
Асинхронный транспорт Cassandra
Вадим Цесько
Одноклассники
HL
HighLoad++
2022
Яндекс

Одноклассники



Одноклассники

- **Техническая платформа VK**[a]



Одноклассники

- **Техническая платформа VK**[a]
- $>$ **12K серверов**
- **7 ЦОДов**



Одноклассники

- **Техническая платформа VK**[a]
- $>$ **12K серверов**
- **7 ЦОДов**
- $>$ **4 Тб/с**



Одноклассники

- **Техническая платформа VK**[a]
- $>$ **12K серверов**
- **7 ЦОДов**
- $>$ **4 Тб/с**
- $>$ **100K rps** на ноду (4 ядра)
- **p99** $<$ **100 мс**



Одноклассники

- **Техническая платформа VK**[a]
- $>$ **12K серверов**
- **7 ЦОДов**
- $>$ **4 Тб/с**
- $>$ **100K rps** на ноду (4 ядра)
- **p99** $<$ **100 мс**
- $>$ **1 ЭБ**

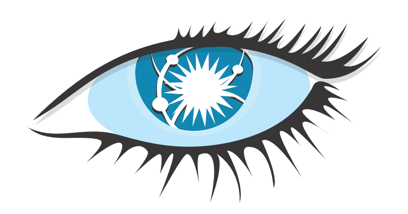
[a] VK Видео, VK Звонки, RuStore и др.

Одноклассники

- **Техническая платформа VK**[a]
- $>$ **12K серверов**
- **7 ЦОДов**
- $>$ **4 Тб/с**
- $>$ **100K rps** на ноду (4 ядра)
- **p99** $<$ **100 мс**
- $>$ **1 ЭБ**

[a] VK Видео, VK Звонки, RuStore и др.

Cassandra[1] в OK[2]

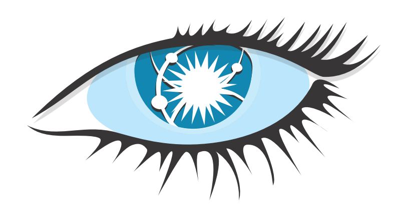


[1] Олег Анастасьев. NewSQL = NoSQL+ACID

[2] Олег Анастасьев. Эффективные надежные микросервисы



Cassandra[1] в OK[2]



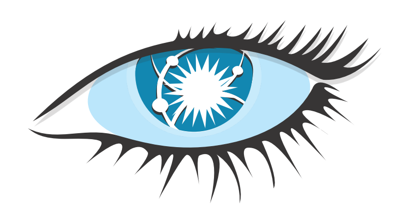
- $>$ **100 кластеров**

[1]Олег Анастасьев. NewSQL = NoSQL+ACID

[2]Олег Анастасьев. Эффективные надежные микросервисы



Cassandra[1] в OK[2]



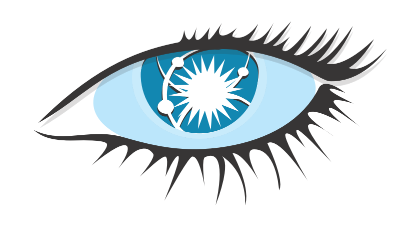
- $>$ **100 кластеров**
- **До 600 нод/кластер**

[1]Олег Анастасьев. NewSQL = NoSQL+ACID

[2]Олег Анастасьев. Эффективные надежные микросервисы



Cassandra[1] в ОК[2]



- $>$ **100 кластеров**
- **До 600 нод/кластер**
- $>$ **1K клиентов/кластер**

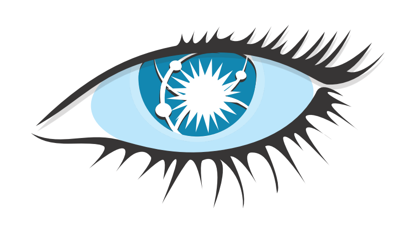
---

[1] Олег Анастасьев. NewSQL = NoSQL+ACID

[2] Олег Анастасьев. Эффективные надежные микросервисы



Cassandra[1] в OK[2]



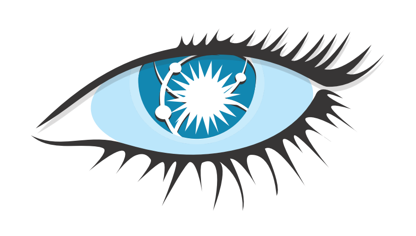
- $>$ 100 кластеров
- До 600 нод/кластер
- $>$ 1K клиентов/кластер
- До 150K rps/ноду

[1] Олег Анастасьев. NewSQL = NoSQL+ACID

[2] Олег Анастасьев. Эффективные надежные микросервисы



Cassandra[1] в OK[2]



- $>$ **100 кластеров**
- **До 600 нод/кластер**
- $>$ **1K клиентов/кластер**
- **До 150K rps/ноду**
- **Спекулятивные** запросы

[1] Олег Анастасьев. NewSQL = NoSQL+ACID

[2] Олег Анастасьев. Эффективные надежные микросервисы



N0

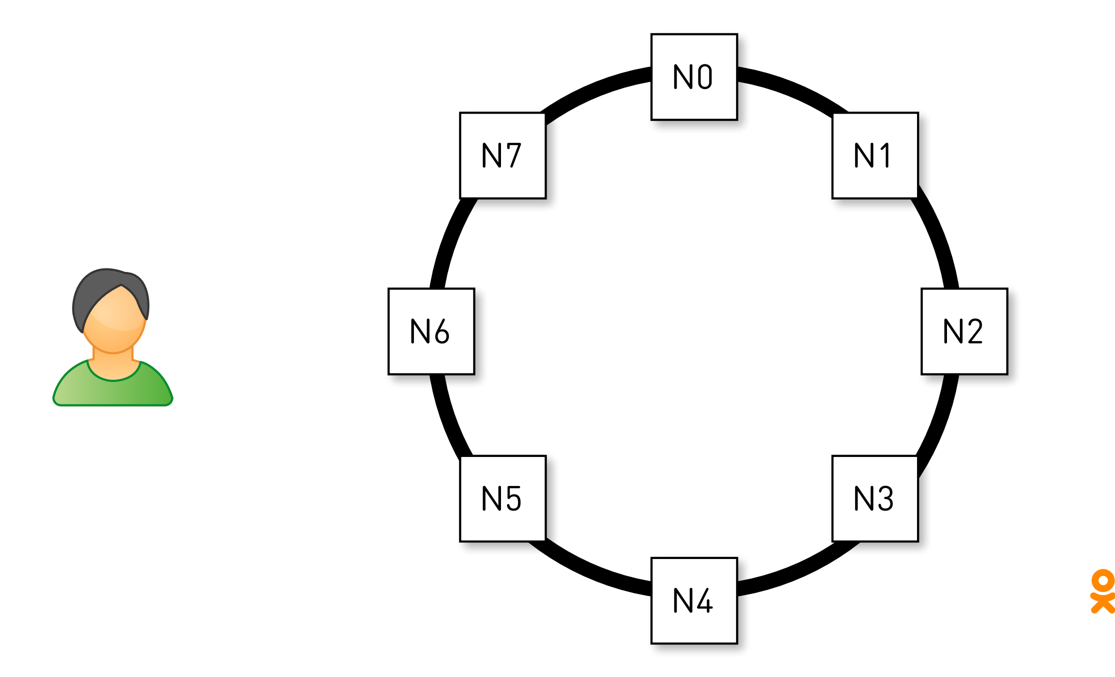
N7 N1

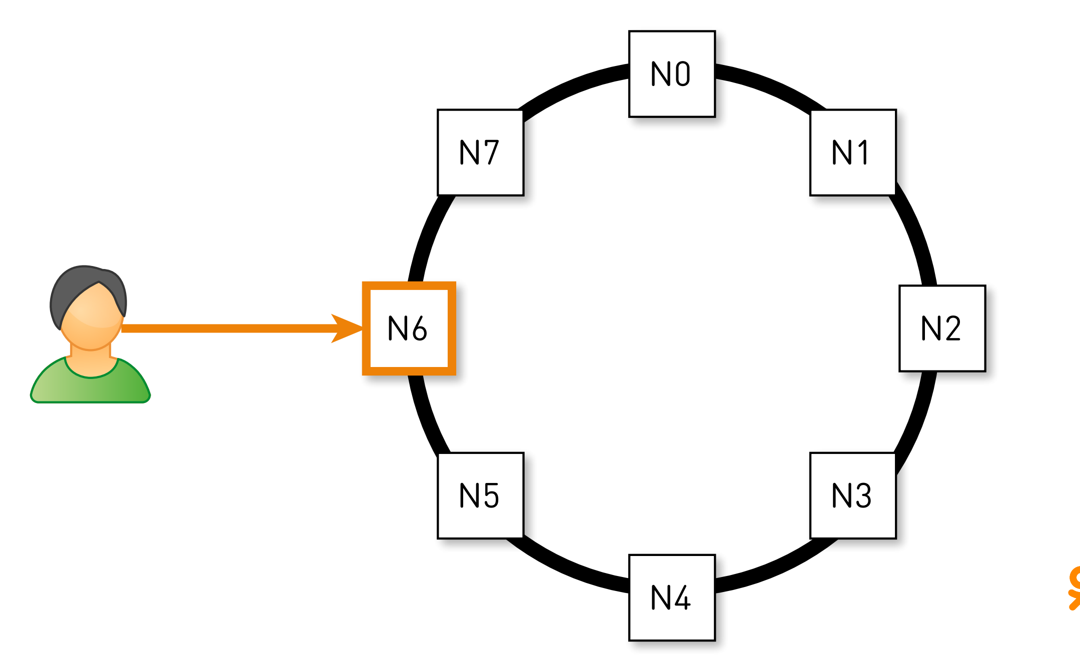
N6 N2

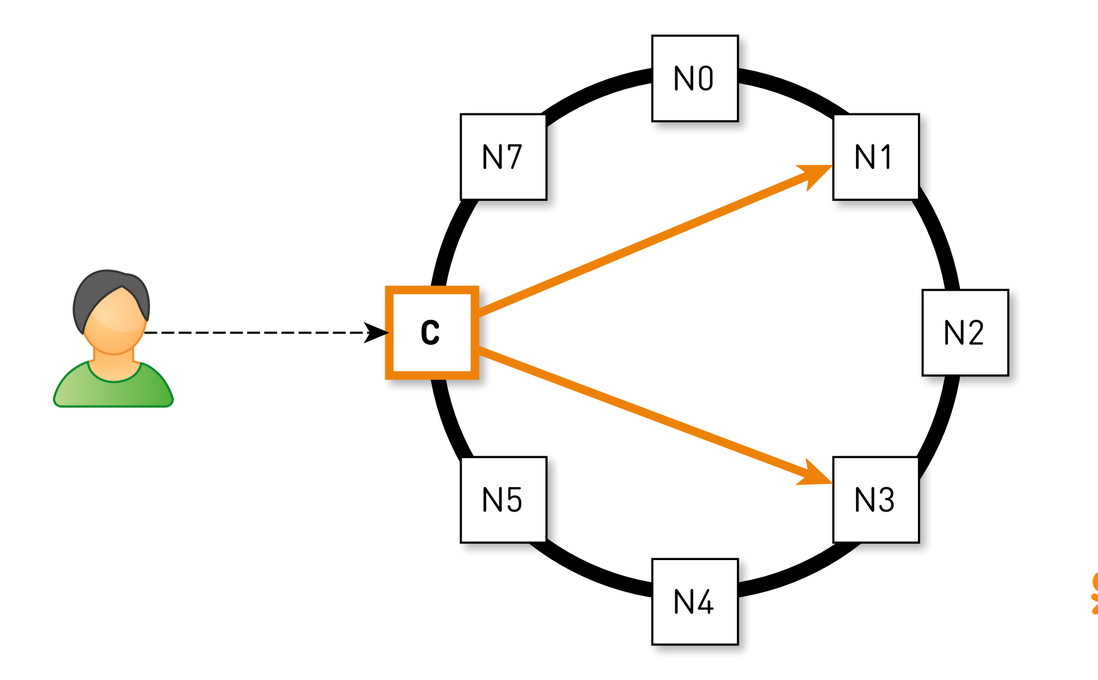
N5 N3

N4



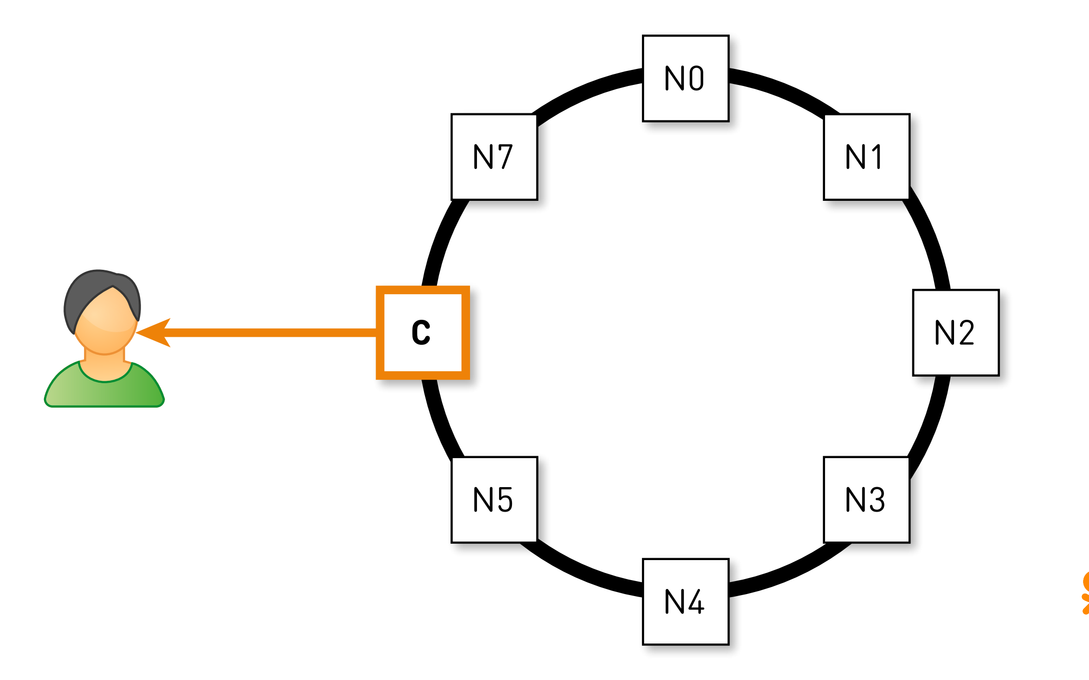

N0
N1
N2
N3
N4
N5
N6
N7

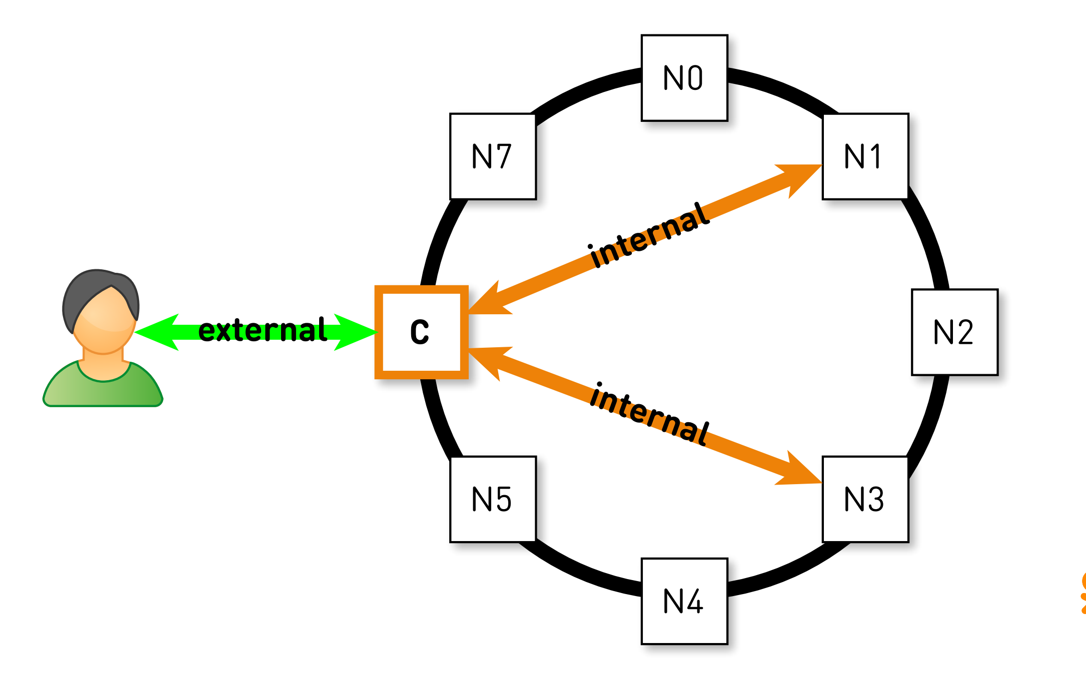

N0
N1
N2
N3
N4
N5
N6
N7


N0
N7
N1
C
N2
N5
N3
N4

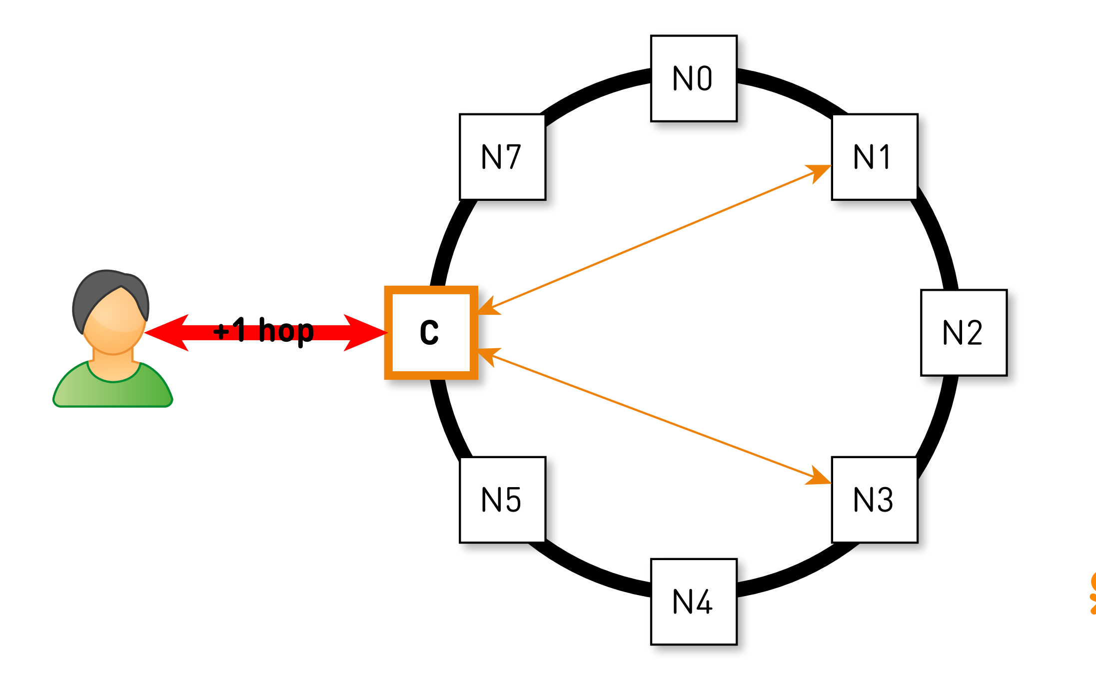
N0
N7
N1
C
N2
N5
N3
N4

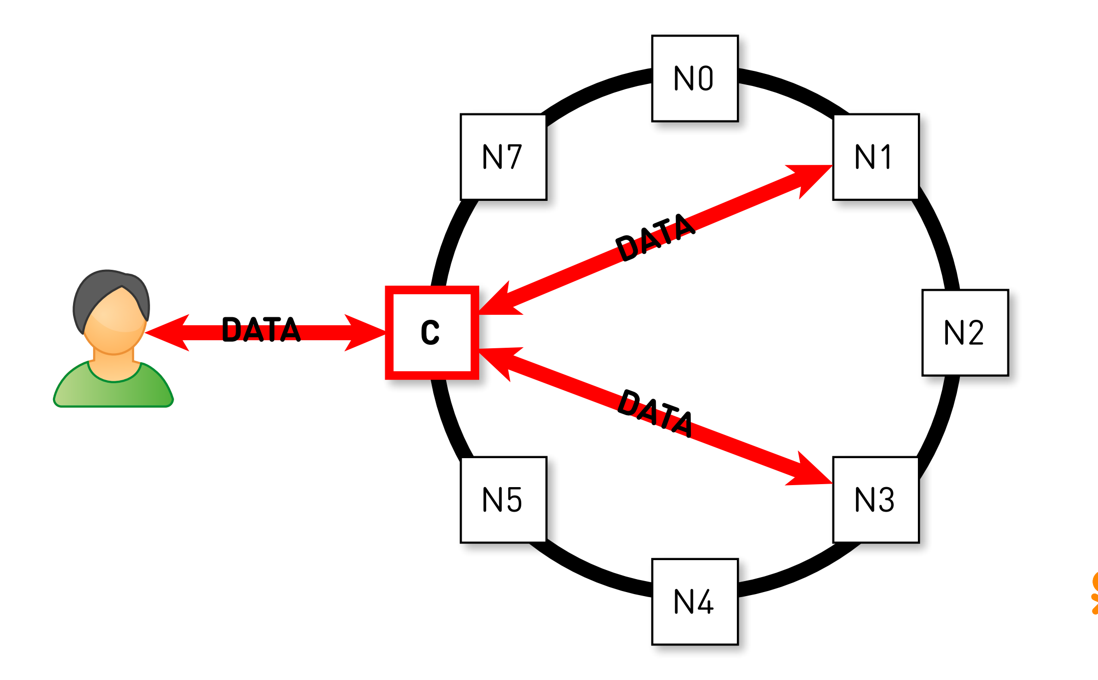


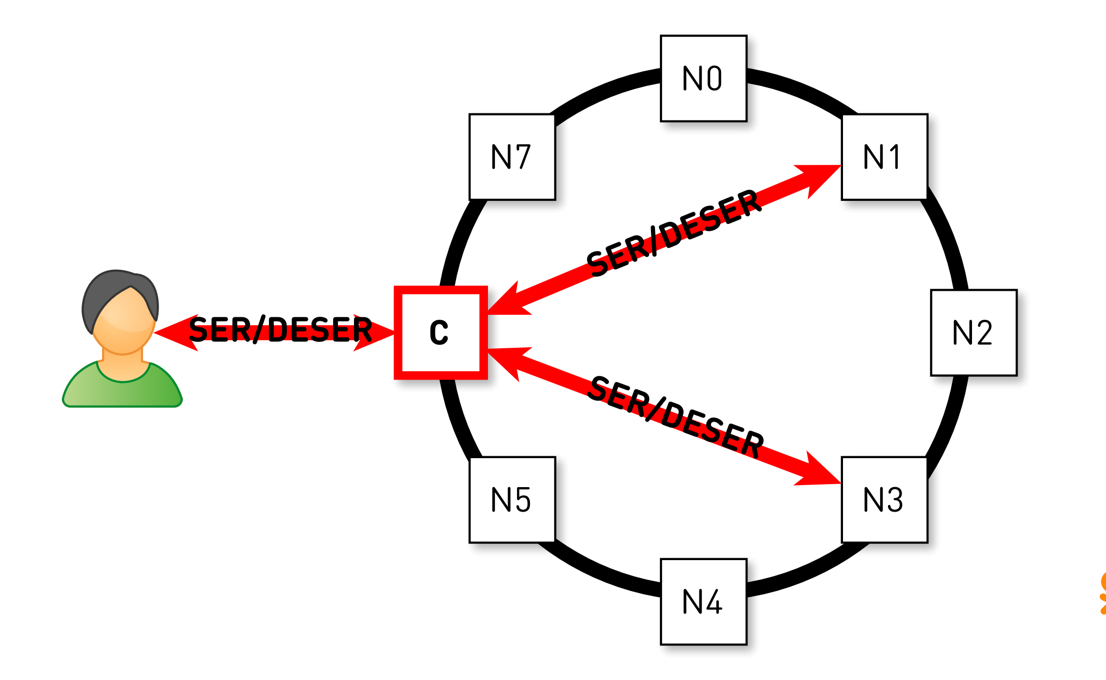

N0
N7
N1
C
N2
N5
N3
N4

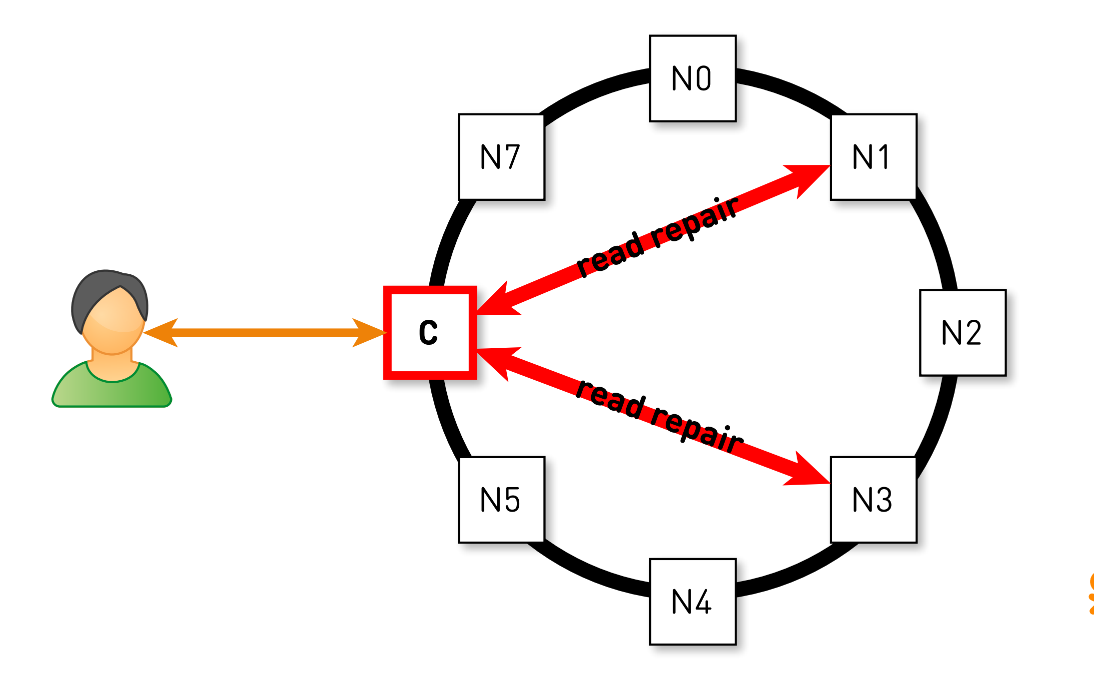


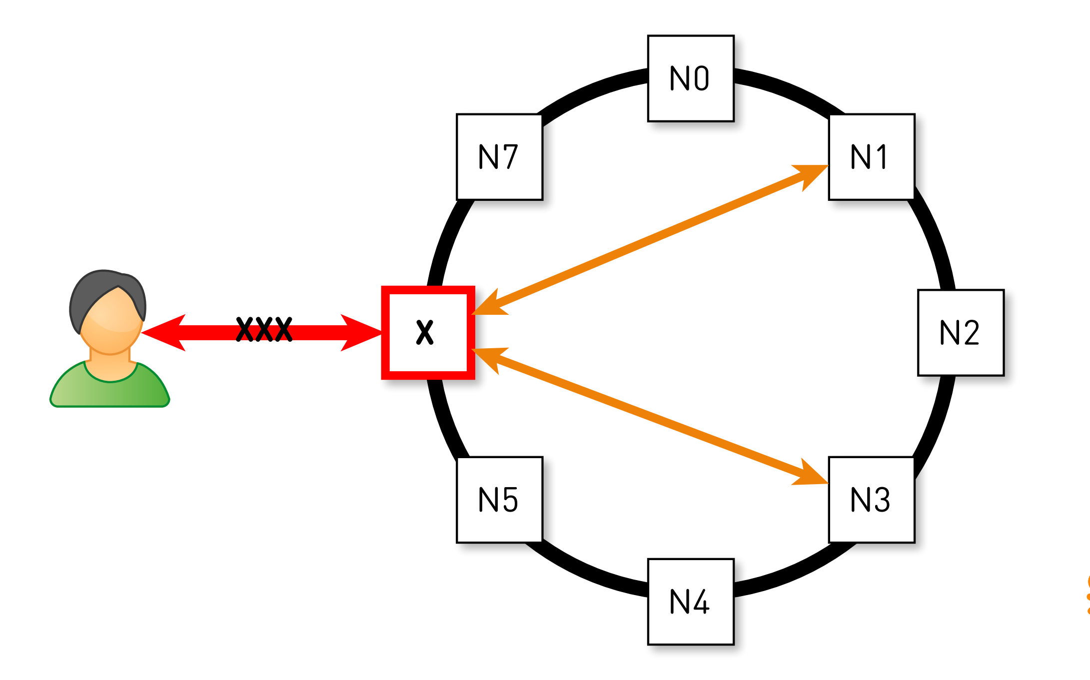
N0
N7
N1
internal
external
C
N2
internal
N5
N3
N4


N0
N7
N1
+1 hop
C
N2
N5
N3
N4

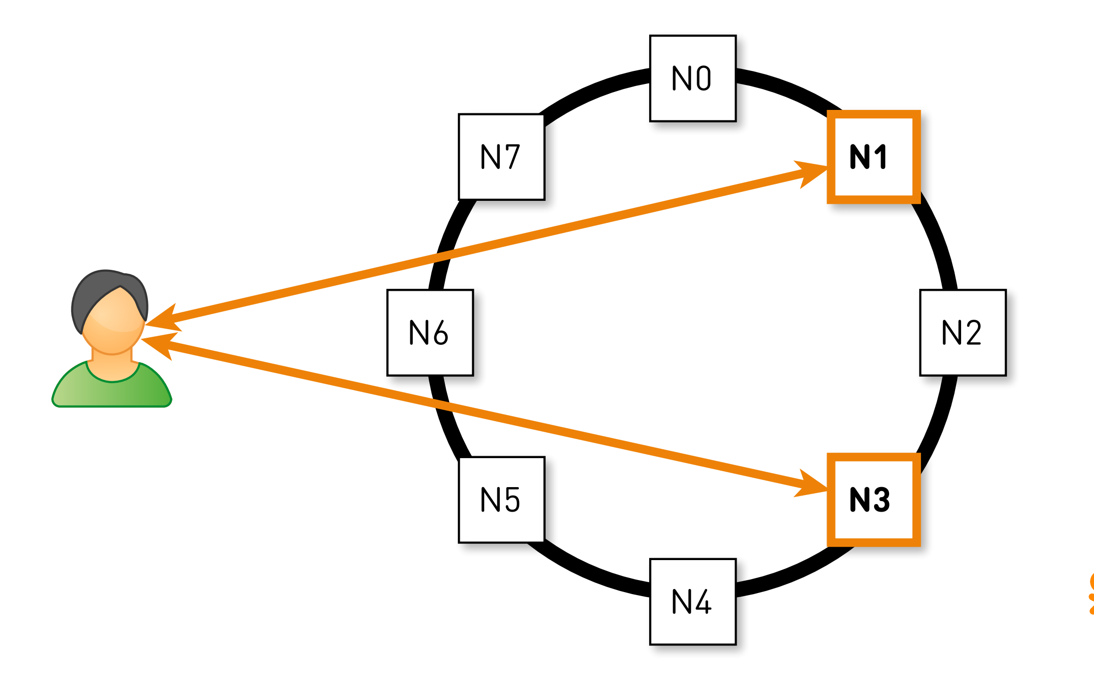

N0
N7
N1
DATA
C
N2
DATA
DATA
N5
N3
N4

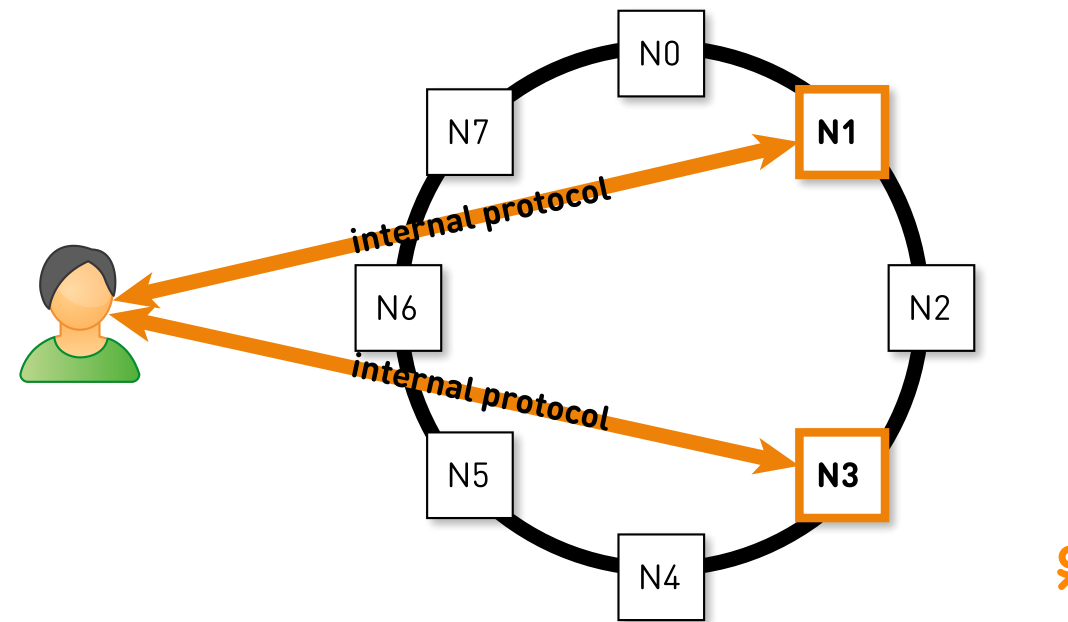

N0
N7
N1
SER/DESER
SER/DESER
C
N2
SER/DESER
N5
N3
N4

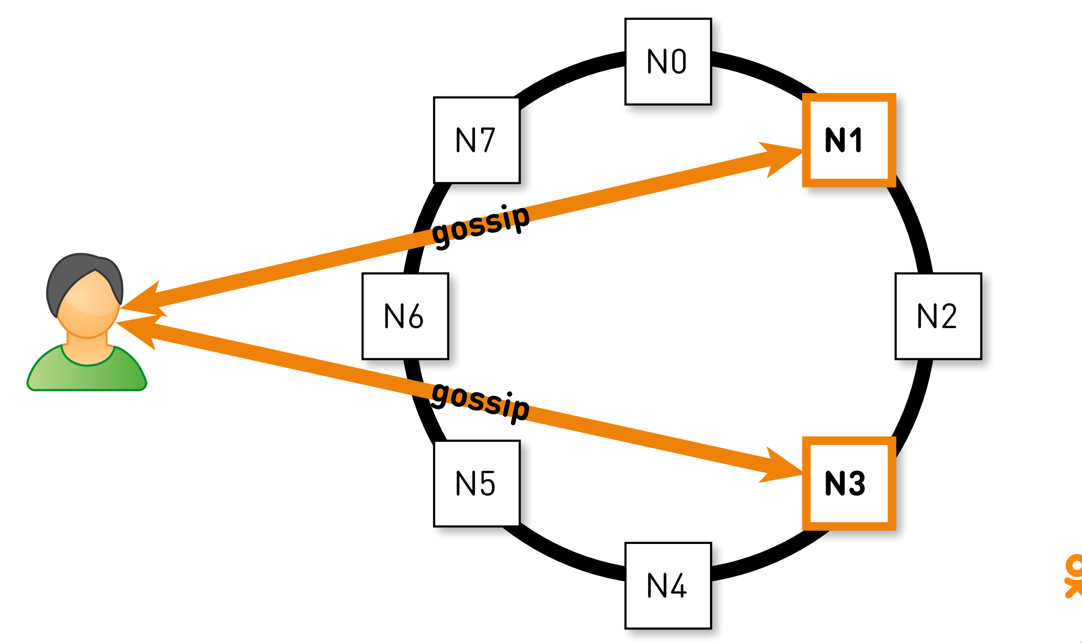

N0
N7
N1
read repair
C
N2
read repair
N5
N3
N4

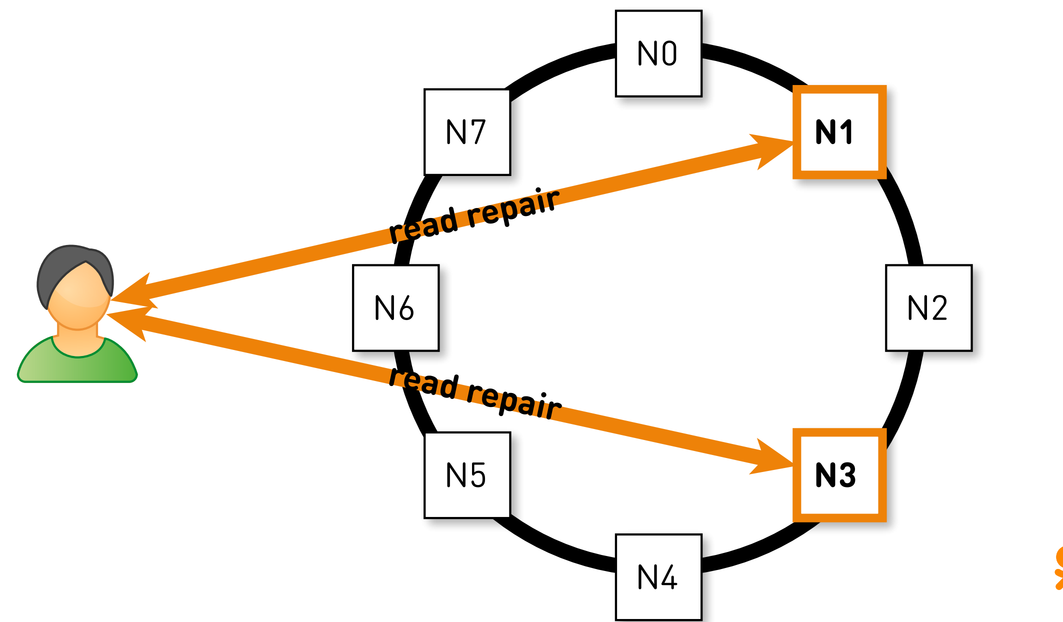


XXX
X
N0
N1
N2
N3
N4
N5
N7


N0
N1
N2
N3
N4
N5
N6
N7


N0
N7
N1
internal protocol
N6
N2
internal protocol
N5
N3
N4


N0
N7
N1
gossip
N6
N2
gossip
N5
N3
N4


N0
N7
N1
read repair
N6
N2
read repair
N5
N3
N4

N0

N7

N1

N6

N2

N5

N3

N4



«Толстый» клиент

- Клиент — **участник** кластера
- Просто не хранит данные
- Общается по **внутреннему протоколу** C*
- Участвует в **Gossip**
- **Координация** на клиентах
- ...



«Толстый» клиент

- Клиент — **участник** кластера
- Просто не хранит данные
- Общается по **внутреннему протоколу** C*
- Участвует в **Gossip**
- **Координация** на клиентах
- ...
- Но есть нюансы



N0
N7
N1
N6
Cluster A
N2
N5
N3
N4

N0
N7
N1
N6
Cluster B
N2
N5
N3
N4



Singletons everywhere

```
private static class MSHandle {
    public static final IMessagingService instance =
            new MessagingService();
}

public static IMessagingService instance() {
    return MSHandle.instance;
}
```



Singletons everywhere

```java
public static final Schema instance = new Schema();
```



Singletons everywhere

```
public static final Schema instance = new Schema();
```

```
public static final Gossiper instance = new Gossiper();
```



Singletons everywhere

```
public static final Schema instance = new Schema();
```

```
public static final Gossiper instance = new Gossiper();
```

```
public static final IFailureDetector instance =
        new FailureDetector();
```



Singletons everywhere

```java
public static final Schema instance = new Schema();
```

```java
public static final Gossiper instance = new Gossiper();
```

```java
public static final IFailureDetector instance =
        new FailureDetector();
```

```java
public static final StorageProxy instance =
        new StorageProxy();
```



Singletons everywhere

```
public static final Schema instance = new Schema();
```

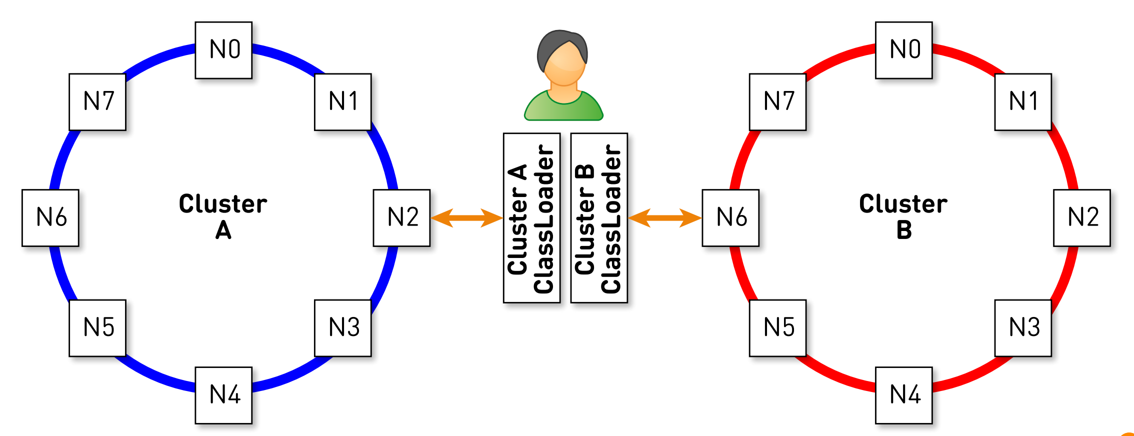
```
public static final Gossiper instance = new Gossiper();
```

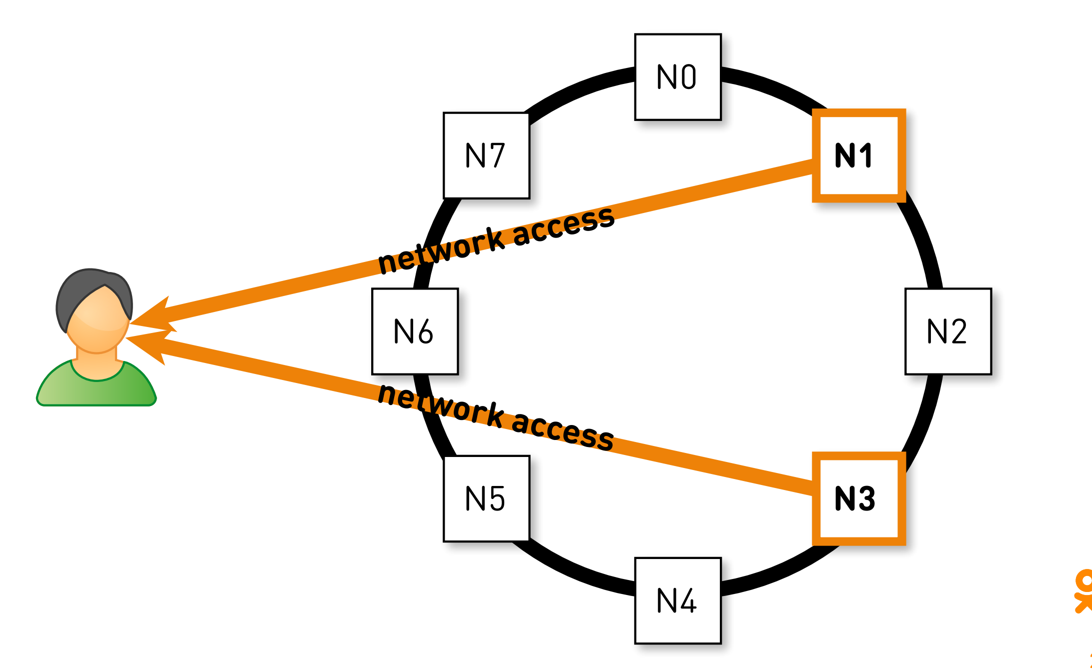
```
public static final IFailureDetector instance =
        new FailureDetector();
```

```
public static final StorageProxy instance =
        new StorageProxy();
```

```
public static final StorageService instance =
        new StorageService();
```

ClassLoader per cluster




N0
N7
N1
network access
N6
N2
network access
N5
N3
N4

Внутренний транспорт

Node 1

Messaging Service

Connection

Node 2

Messaging Service



Односторонние соединения

Node 1

Messaging Service

Outbound

Outbound

Node 2

Messaging Service



ACK и CMD

Node 1
Messaging Service
ACK
CMD
ACK
CMD
Node 2
Messaging Service



Thread per connection

Node 1

Messaging Service

OutboundTcpConnection

OutboundTcpConnection

IncomingTcpConnection

IncomingTcpConnection

ACK

CMD

ACK

CMD

Node 2

IncomingTcpConnection

IncomingTcpConnection

OutboundTcpConnection

OutboundTcpConnection

Messaging Service



Клиент — тоже участник кластера



Итого

- Каждый **клиент**: по **4 потока** для **каждой ноды**
- Каждая **нода**: по **4 потока** для **каждого клиента** и **каждой ноды**



Итого

- Каждый **клиент**: по **4 потока** для **каждой ноды**
- Каждая **нода**: по **4 потока** для **каждого клиента** и **каждой ноды**
- **Тысячи** соединений/**потоков** в каждом процессе



Итого

- Каждый **клиент**: по **4 потока** для **каждой ноды**
- Каждая **нода**: по **4 потока** для **каждого клиента** и **каждой ноды**
- **Тысячи** соединений/**потоков** в каждом процессе
- Все соединения/потоки **активны**



Итого

- Каждый **клиент**: по **4 потока** для **каждой ноды**
- Каждая **нода**: по **4 потока** для **каждого клиента** и **каждой ноды**
- **Тысячи** соединений/**потоков** в каждом процессе
- Все соединения/потоки **активны**
- И сетевой доступ в **обе стороны**



Цели



Цели

- Эксплуатация
    - Соединения **`client`** $\rightarrow$ **`node`** и `node` $\leftrightarrow$ `node`



Цели

- Эксплуатация
  - Соединения **`client`** $\rightarrow$ **`node`** и `node` $\leftrightarrow$ `node`
  - Поддержать **клиентов** за NAT/VPN



Цели

- Эксплуатация
  - Соединения **`client`** $\to$ **`node`** и `node` $\leftrightarrow$ `node`
  - Поддержать **клиентов** за NAT/VPN
- Оптимизация
  - Обслуживать **небольшим** пулом потоков



Задачи



Задачи

- Перейти на **Async IO**



Задачи

- Перейти на **Async IO**
- Использовать **входящие соединения**
- Передавать данные **в обе стороны**



MessagingService

```java
void listen(InetAddress localEp);

void receive(MessageIn message);

void sendOneWay(MessageOut message, InetAddress to);

void sendReply(MessageIn request, MessageOut response);

void shutdown();
```

MessagingService

```
void listen(InetAddress localEp);

void receive(MessageIn message);

void sendOneWay(MessageOut message, InetAddress to);

void sendReply(MessageIn request, MessageOut response);

void shutdown();
```

MessagingService

```java
void listen(InetAddress localEp);

void receive(MessageIn message);

void sendOneWay(MessageOut message, InetAddress to);

void sendReply(MessageIn request, MessageOut response);

void shutdown();
```

MessagingService

```
void listen(InetAddress localEp);

void receive(MessageIn message);

void sendOneWay(MessageOut message, InetAddress to);

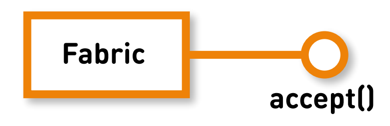
void sendReply(MessageIn request, MessageOut response);

void shutdown();
```


Messaging Service
Fabric



Messaging Service





Messaging Service

Fabric

accept()

Node 10.0.0.1

Node 10.0.0.2

Node ...

...

Connection

Connection

Connection



Messaging Service

Fabric

accept()

Node 10.0.0.1

Node 10.0.0.2

Node ...

...

Connection

Connection

Connection

In

Out

In

Out

socket

socket



Messaging Service

Fabric

accept()

Node 10.0.0.1

Node 10.0.0.2

Node ...

Connection ...

Connection ACK

Connection CMD

Connection ...

In

Out

In

Out

socket

socket



Messaging Service

Fabric

accept()

Node 10.0.0.1

Node 10.0.0.2

Node ...

Connection ...

Connection ACK

Connection CMD

Connection ...

In

Out

In

Out

socket

socket



AsynchronousSocketChannel

```
public abstract <A> void connect(
    SocketAddress remote,
    A attachment,
    CompletionHandler<Void, ? super A> handler);

public abstract <A> void read/write(
    ByteBuffer dst/src,
    long timeout,
    TimeUnit unit,
    A attachment,
    CompletionHandler<Integer, ? super A> handler);
```

CompletionHandler

```
void completed(V result, A attachment);

void failed(Throwable exc, A attachment);
```



```
completed(V result, A attachment)
```



`completed(V result, A attachment)`

- Завершилась ли предыдущая запись/чтение?



completed(V result, A attachment)

- Завершилась ли предыдущая запись/чтение?
- Полностью ли записали/прочитали?



completed(V result, A attachment)

- Завершилась ли предыдущая запись/чтение?
- Полностью ли записали/прочитали?
- Можно начинать парсить/отправлять следующее сообщение?



`completed(V result, A attachment)`

- Завершилась ли предыдущая запись/чтение?
- Полностью ли записали/прочитали?
- Можно начинать парсить/отправлять следующее сообщение?
- В каком потоке вызвался callback?



completed(V result, A attachment)

- Завершилась ли предыдущая запись/чтение?
- Полностью ли записали/прочитали?
- Можно начинать парсить/отправлять следующее сообщение?
- В каком потоке вызвался callback?
- Что и от кого нужно защищать?



`completed(V result, A attachment)`

- Завершилась ли предыдущая запись/чтение?
- Полностью ли записали/прочитали?
- Можно начинать парсить/отправлять следующее сообщение?
- В каком потоке вызвался callback?
- Что и от кого нужно защищать?
- ...



CompletionHandler

```
void completed(V result, A attachment);

void failed(Throwable exc, A attachment);
```



Messaging Service

Fabric

accept()

Node 10.0.0.1

Node 10.0.0.2

Node ...

Connection ...

Connection ACK

Connection CMD

Connection ...

In

Out

In

Out

socket

socket



Messaging Service

Fabric

accept()

Node 10.0.0.1

Node 10.0.0.2

Node ...

Connection ...

Connection ACK

Connection CMD

Connection ...

In

Out

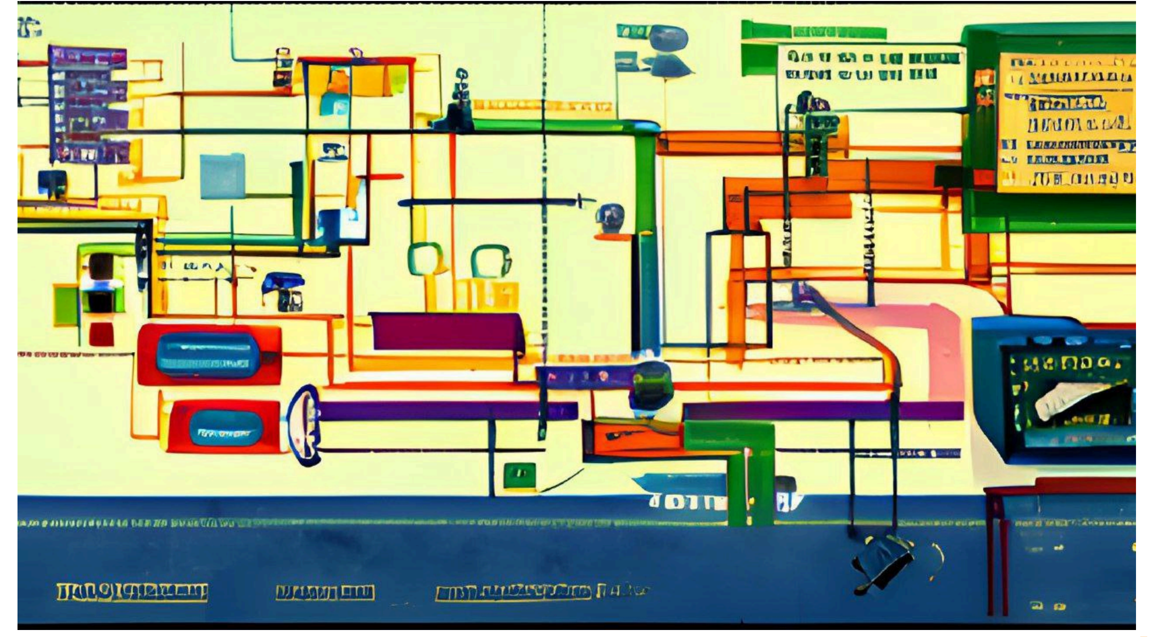
In

Out

socket

socket





Что мы хотим?

- **Простой последовательный** код



Что мы хотим?

- **Простой последовательный** код
- Обслуживать **тысячи соединений**
- На **небольшом пуле** потоков

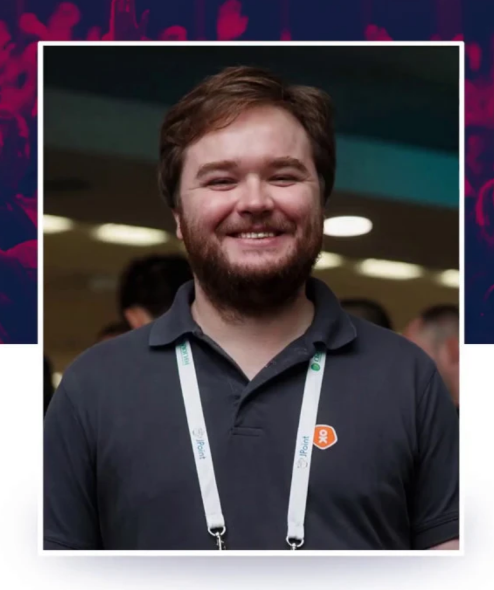


Joker<?> 2018

Вадим Цесько

Одноклассники

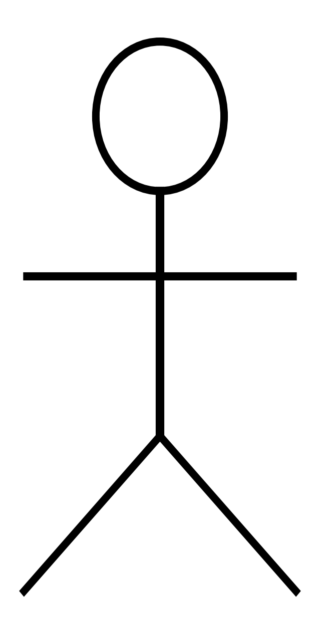
Реактивный раздатчик
ok.ru/music



Actor Model в помощь

Proxy client
Publisher
Security check
IDv3 tagging
External API
Pub
Sub
Pub
Sub
Router
Internal API
Caching storage
Pub
Sub
Storage
Disk 0
Disk ...
Disk N









1



	2	1



3	2	1



3	2	1



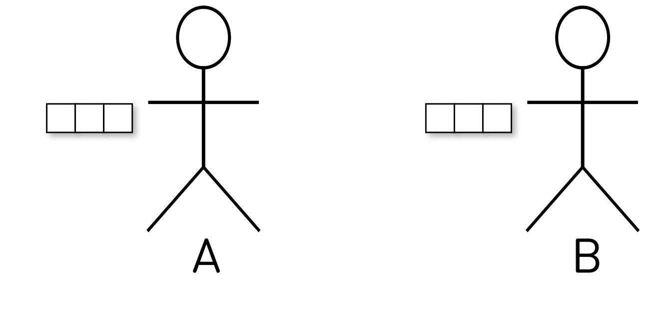
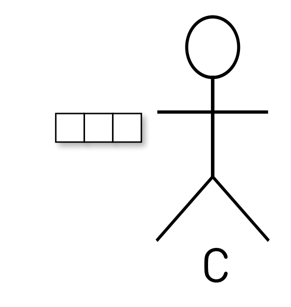
	3	**2**

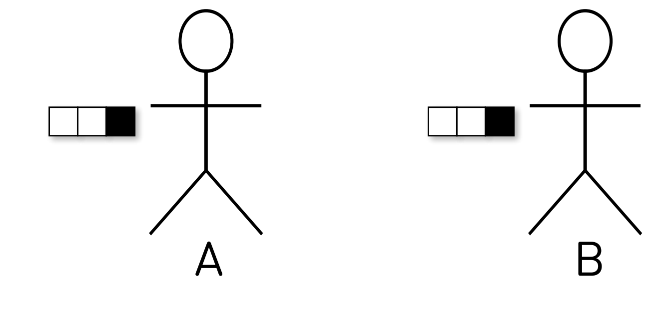
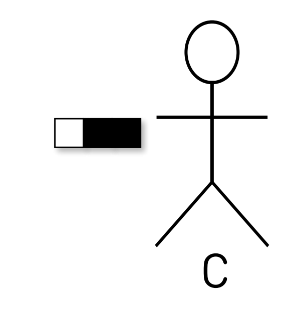


3






A
B

C

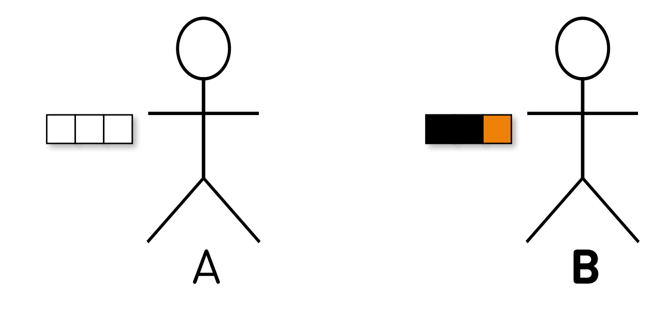


A
B

C

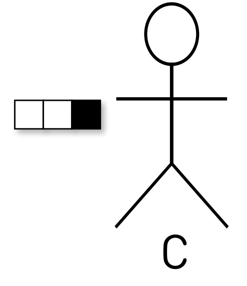

A

B

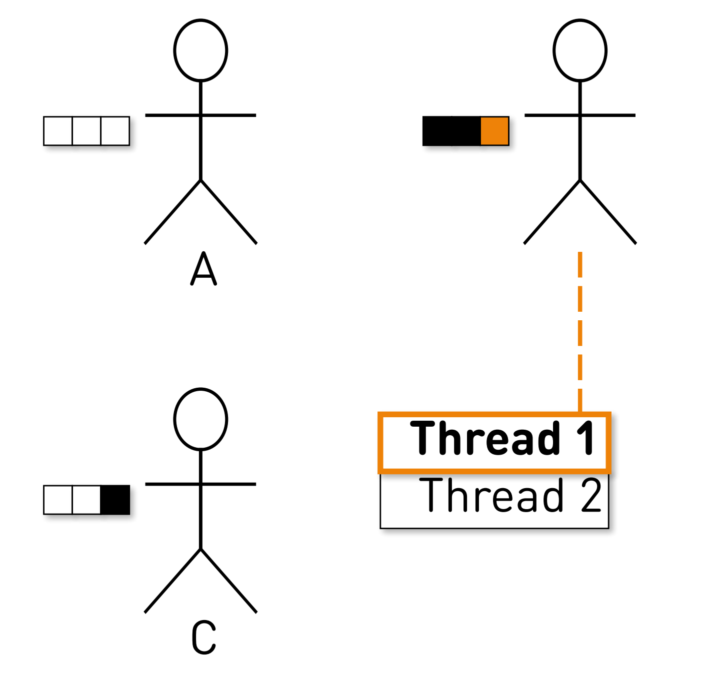
C




A
B


C




A
Thread 1
Thread 2
C

A

C

Thread 1

Thread 2



A

C

Thread 1

Thread 2



A

C

C

Thread 1

Thread 2



Ссылки

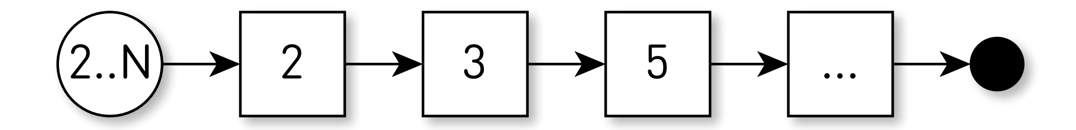
- **Потоковая обработка данных с помощью Actor Model** @ ADD 2012
- **Akka в Яндекс** @ JPoint 2014
- **Реактивный раздатчик ok.ru/music** @ Joker 2018



one-actor

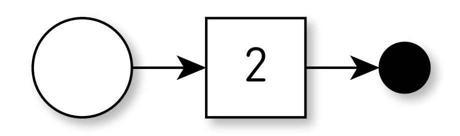
```
/**
 * Implements sieve of Eratosthenes.
 * <p>
 * Allocates a child {@link Divisor} when a new prime is found.
 * Sifts candidates to the child if can't divide them.
 * <p>
 * Supports recursive {@link #collect(Collection)} request which
 * {@link #stop()}s all the children in the chain starting from
 * the last item.
 *
 * @author incubos
 */
class Divisor extends Actor<Divisor.Message> {
```

Решето Эратосфена



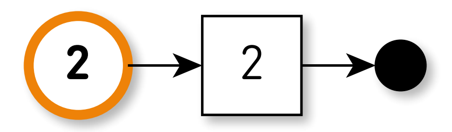


Инициализация



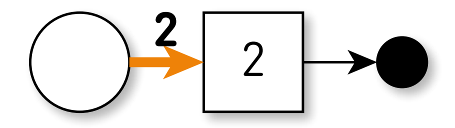


Просеиваем 2



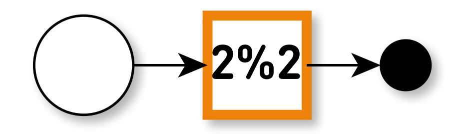


Просеиваем 2



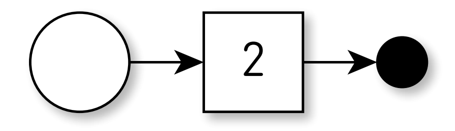


Просеиваем 2



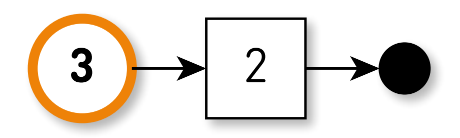


Отсеяли



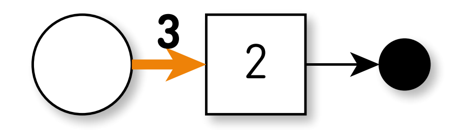


Просеиваем 3



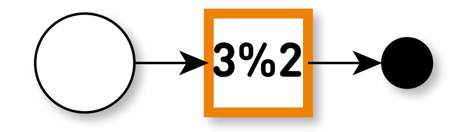


Просеиваем 3



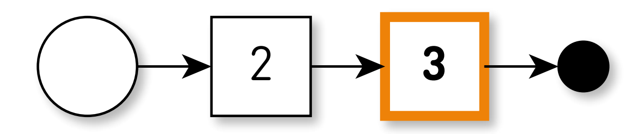


Просеиваем 3



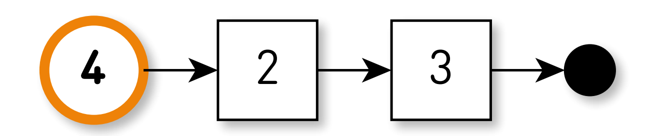


Просеяли



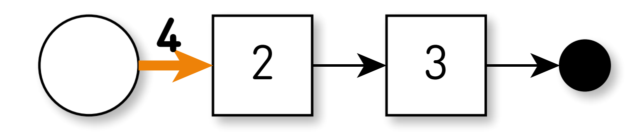


Просеиваем 4



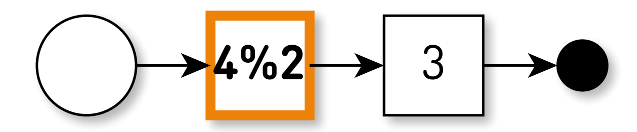


Просеиваем 4



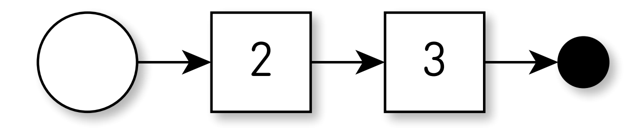


Просеиваем 4



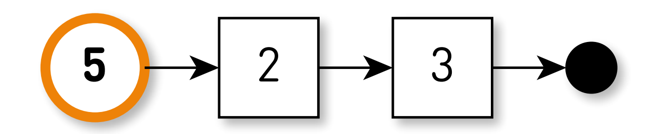


Отсеяли



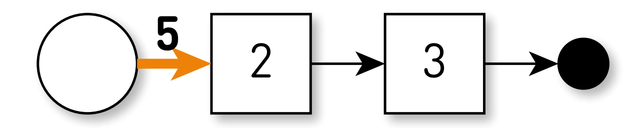


Просеиваем 5



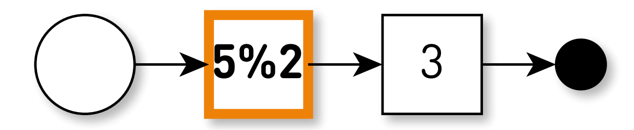


Просеиваем 5



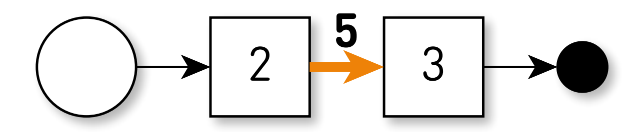


Просеиваем 5



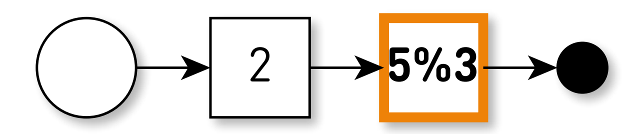


Просеиваем 5



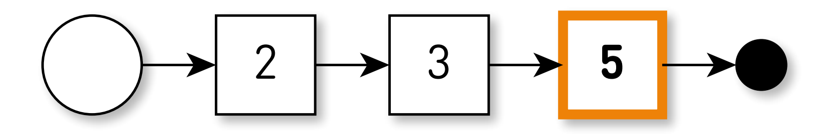


Просеиваем 5



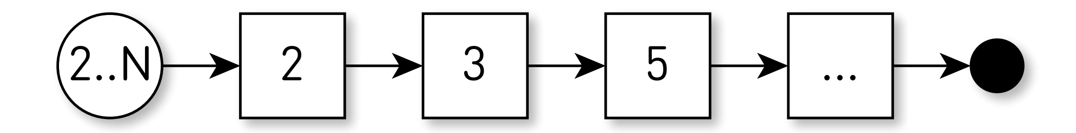


Просеяли





И так далее





Состояние

```
class Divisor extends Actor<Divisor.Message> {
    final long prime;
    Divisor next = null;
```



Состояние

```
class Divisor extends Actor<Divisor.Message> {
    final long prime;
    Divisor next = null;
```



Состояние

```
class Divisor extends Actor<Divisor.Message> {
    final long prime;
    Divisor next = null;
```



Конструирование

```
Divisor(
        Supervisor parent,
        long prime,
        ExecutorService executor) {
    super(parent, prime, executor);

    this.prime = prime;
}
```



Конструирование

```
Divisor(
        Supervisor parent,
        long prime,
        ExecutorService executor) {
    super(parent, prime, executor);

    this.prime = prime;
}
```



Конструирование

```
Divisor(
        Supervisor parent,
        long prime,
        ExecutorService executor) {
    super(parent, prime, executor);

    this.prime = prime;
}
```



Конструирование

```
Divisor(
        Supervisor parent,
        long prime,
        ExecutorService executor) {
    super(parent, prime, executor);

    this.prime = prime;
}
```



Supervisor

```java
void onChildError(Actor<?> child, Exception e);

void onChildStop(Actor<?> child);
```



Supervisor

```
void onChildError(Actor<?> child, Exception e);

void onChildStop(Actor<?> child);
```



Supervisor

```
void onChildError(Actor<?> child, Exception e);

void onChildStop(Actor<?> child);
```



Death Pact

```java
@Override
public void onChildStop(Actor<?> child) {
    assert child == next;

    stop();
}
```



Protocol

```
static final class Check extends Message {
    final long candidate;

    Check(long candidate) { ... }
}
```



Protocol

```java
static final class Check extends Message {
    final long candidate;

    Check(long candidate) { ... }
}
```

```java
static final class Collect extends Message {
    final Collection<Long> to;

    Collect(Collection<Long> to) { ... }
}
```



Matching

```java
@Override
protected void receive(Message message) {
    switch (message) {
        case Check check -> doCheck(check.candidate);
        case Collect collect -> doCollect(collect.to);
        default -> throw new IllegalArgumentException();
    }
}
```



```java
void doCheck(long candidate) {
    if (candidate % prime == 0) {
        // Not prime
        return;
    }

    if (next == null) {
        // New prime discovered
        next = new Divisor(this, candidate, dispatcher);
    } else {
        // Pass through
        next.check(candidate);
    }
}
```

```java
void doCheck(long candidate) {
    if (candidate % prime == 0) {
        // Not prime
        return;
    }

    if (next == null) {
        // New prime discovered
        next = new Divisor(this, candidate, dispatcher);
    } else {
        // Pass through
        next.check(candidate);
    }
}
```

```java
void doCheck(long candidate) {
    if (candidate % prime == 0) {
        // Not prime
        return;
    }

    if (next == null) {
        // New prime discovered
        next = new Divisor(this, candidate, dispatcher);
    } else {
        // Pass through
        next.check(candidate);
    }
}
```

```java
void doCheck(long candidate) {
    if (candidate % prime == 0) {
        // Not prime
        return;
    }

    if (next == null) {
        // New prime discovered
        next = new Divisor(this, candidate, dispatcher);
    } else {
        // Pass through
        next.check(candidate);
    }
}
```

```java
void doCollect(Collection<Long> to) {
    to.add(prime);
    if (next == null) {
        stop();
    } else {
        next.collect(to);
    }
}
```

```java
void doCollect(Collection<Long> to) {
    to.add(prime);
    if (next == null) {
        stop();
    } else {
        next.collect(to);
    }
}
```

```java
void doCollect(Collection<Long> to) {
    to.add(prime);
    if (next == null) {
        stop();
    } else {
        next.collect(to);
    }
}
```

```java
void doCollect(Collection<Long> to) {
    to.add(prime);
    if (next == null) {
        stop();
    } else {
        next.collect(to);
    }
}
```

```
var dispatcher = new Dispatcher("eratosthenes");
var stopped = new CountDownLatch(1);
var supervisor = new Supervisor() {
    @Override
    public void onChildError(Actor<?> child, Exception e) {
        onChildStop(child);
    }

    @Override
    public void onChildStop(Actor<?> child) {
        stopped.countDown();
    }
};
var sieve = new Divisor(supervisor, 2, dispatcher);
```

```java
var dispatcher = new Dispatcher("eratosthenes");
var stopped = new CountDownLatch(1);
var supervisor = new Supervisor() {
        @Override
        public void onChildError(Actor<?> child, Exception e) {
                onChildStop(child);
        }

        @Override
        public void onChildStop(Actor<?> child) {
                stopped.countDown();
        }
};
var sieve = new Divisor(supervisor, 2, dispatcher);
```

```
var dispatcher = new Dispatcher("eratosthenes");
var stopped = new CountDownLatch(1);
var supervisor = new Supervisor() {
    @Override
    public void onChildError(Actor<?> child, Exception e) {
        onChildStop(child);
    }

    @Override
    public void onChildStop(Actor<?> child) {
        stopped.countDown();
    }
};
var sieve = new Divisor(supervisor, 2, dispatcher);
```

```java
var dispatcher = new Dispatcher("eratosthenes");
var stopped = new CountDownLatch(1);
var supervisor = new Supervisor() {
    @Override
    public void onChildError(Actor<?> child, Exception e) {
        onChildStop(child);
    }

    @Override
    public void onChildStop(Actor<?> child) {
        stopped.countDown();
    }
};
var sieve = new Divisor(supervisor, 2, dispatcher);
```

Вычисление

```java
for (long i = 3; i <= 1000; i++) {
    sieve.enqueue(new Check(i));
}

Deque<Long> primes = new LinkedBlockingDeque<>();
sieve.enqueue(new Collect(primes));

stopped.await();

assertEquals(997, primes.getLast().longValue());
```

Вычисление

```
for (long i = 3; i <= 1000; i++) {
    sieve.enqueue(new Check(i));
}

Deque<Long> primes = new LinkedBlockingDeque<>();
sieve.enqueue(new Collect(primes));

stopped.await();

assertEquals(997, primes.getLast().longValue());
```

Вычисление

```
for (long i = 3; i <= 1000; i++) {
    sieve.enqueue(new Check(i));
}

Deque<Long> primes = new LinkedBlockingDeque<>();
sieve.enqueue(new Collect(primes));

stopped.await();

assertEquals(997, primes.getLast().longValue());
```

Вычисление

```
for (long i = 3; i <= 1000; i++) {
    sieve.enqueue(new Check(i));
}

Deque<Long> primes = new LinkedBlockingDeque<>();
sieve.enqueue(new Collect(primes));

stopped.await();

assertEquals(997, primes.getLast().longValue());
```

Вычисление

```
for (long i = 3; i <= 1000; i++) {
    sieve.enqueue(new Check(i));
}

Deque<Long> primes = new LinkedBlockingDeque<>();
sieve.enqueue(new Collect(primes));

stopped.await();

assertEquals(997, primes.getLast().longValue());
```

one-actor

400 LOC без зависимостей.

[all classes]

Overall Coverage Summary

Package	Class, %	Method, %	Line, %
all classes	100% (7/ 7)	100% (31/ 31)	100% (133/ 133)

Coverage Breakdown

Package ▲	Class, %	Method, %	Line, %
one.actor	100% (7/ 7)	100% (31/ 31)	100% (133/ 133)



Fabric

Node
Node
...

Connection
Connection
Connection
...

In
Out
In
Out



Fabric

Node

Node

...

Connection

Connection

Connection

...

In

Out

In

Out



Fabric

Node **Node** **...**

Connection Connection Connection ...

In Out In Out



Fabric

Node

Node

...

Connection

Connection

Connection

...

In

Out

In

Out



Fabric

Node | Node | ...

Connection | Connection | Connection | ...

In | **Out** | **In** | **Out**



```
out.write(...,
        new CompletionHandler<Integer, ByteBuffer>() {
            @Override
            public void completed(...) {
                ...
                enqueue(Ready.INSTANCE);
            }

            @Override
            public void failed(...) {
                ...
                enqueue(WriteFailure.INSTANCE);
            }
        });
```

```java
out.write(...,
        new CompletionHandler<Integer, ByteBuffer>() {
            @Override
            public void completed(...) {
                ...
                enqueue(Ready.INSTANCE);
            }

            @Override
            public void failed(...) {
                ...
                enqueue(WriteFailure.INSTANCE);
            }
        });
```

```java
out.write(...,
        new CompletionHandler<Integer, ByteBuffer>() {
            @Override
            public void completed(...) {
                ...
                enqueue(Ready.INSTANCE);
            }

            @Override
            public void failed(...) {
                ...
                enqueue(WriteFailure.INSTANCE);
            }
        });
```

Messaging Service

Messaging Service

JVM

localhost



SendingBenchmark

- **10 M** сообщений по **1 КБ** в одну сторону
- Отправка **пачками по 100 K** сообщений
- **Не больше двух пачек** в полёте



SendingBenchmark

- **10 M** сообщений по **1 КБ** в одну сторону
- Отправка **пачками по 100 К** сообщений
- **Не больше двух пачек** в полёте

$7\ \mu s$

на сообщение



Messaging Service

Fabric

Node 10.0.0.1

Node 10.0.0.2

Node ...

Connection ...

Connection ACK

Connection CMD

Connection ...

In

Out

In

Out

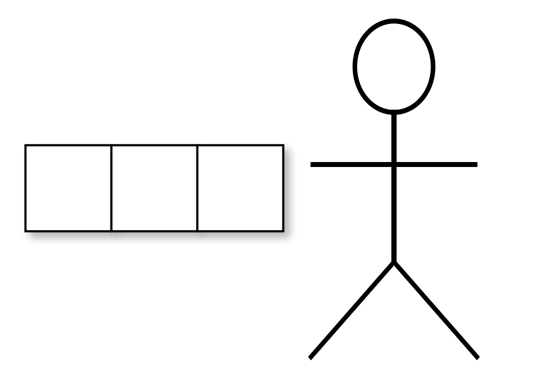
socket

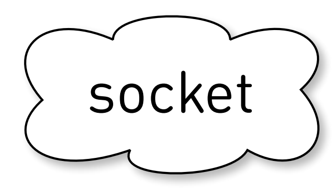
socket



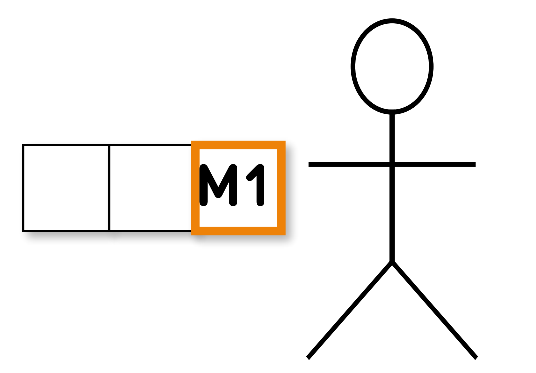
Асинхронная запись

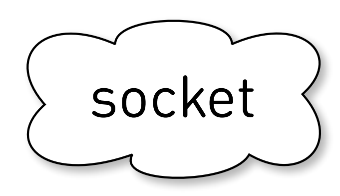
```
/**
 * @throws WritePendingException
 *         If a write operation is already in progress on
 *         this channel
 */
public abstract <A> void write(
       ByteBuffer src,
       long timeout,
       TimeUnit unit,
       A attachment,
       CompletionHandler<Integer,? super A> handler);
```




socket




M1


socket



write(M1)

socket

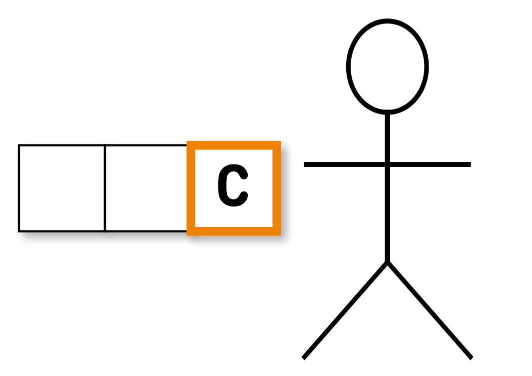


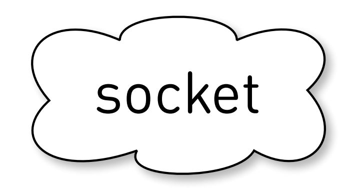
C

completed()

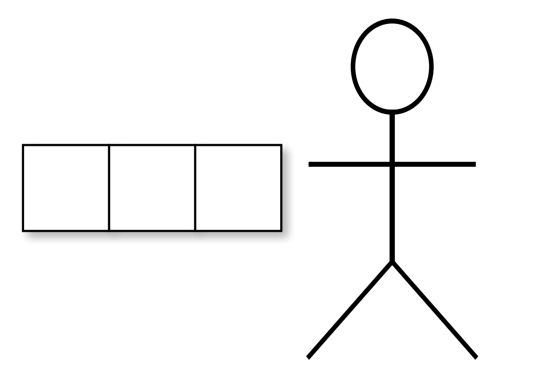
socket

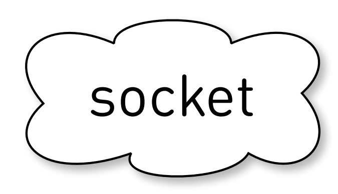



c

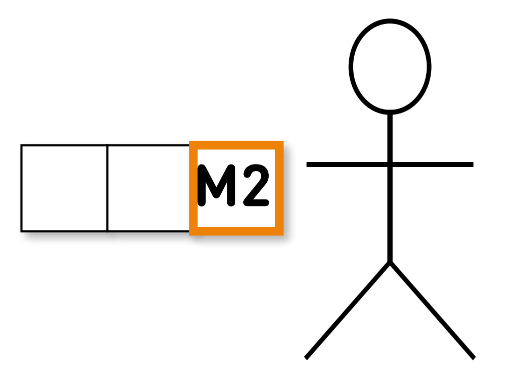

socket

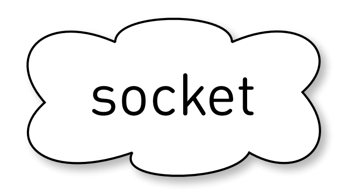





socket




M2

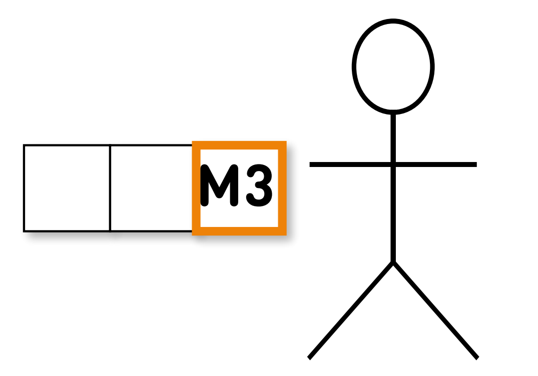

socket

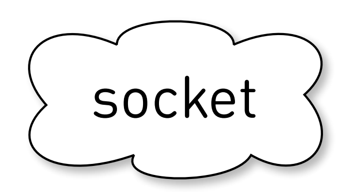


write(M2)

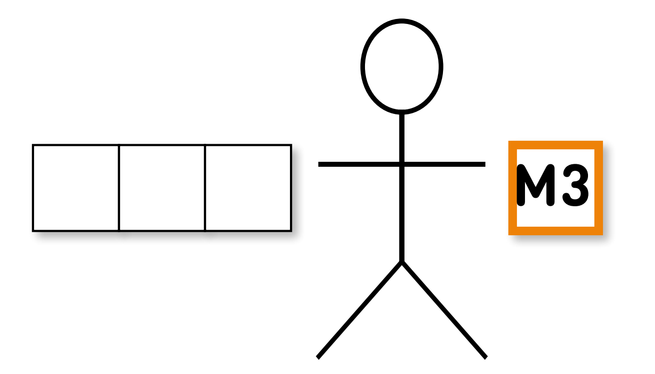
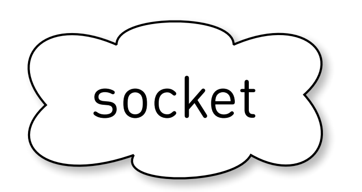
socket

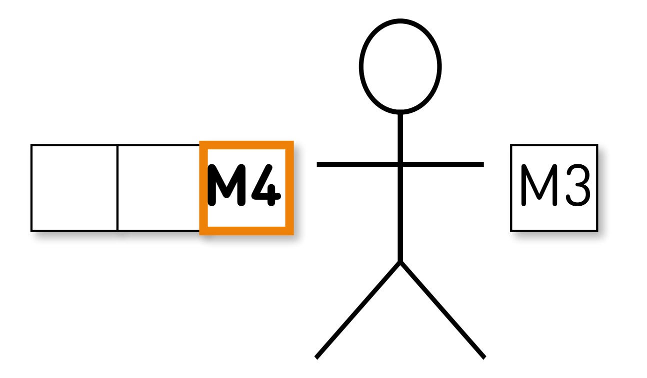


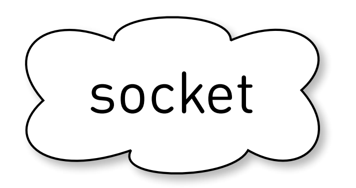

M3


socket

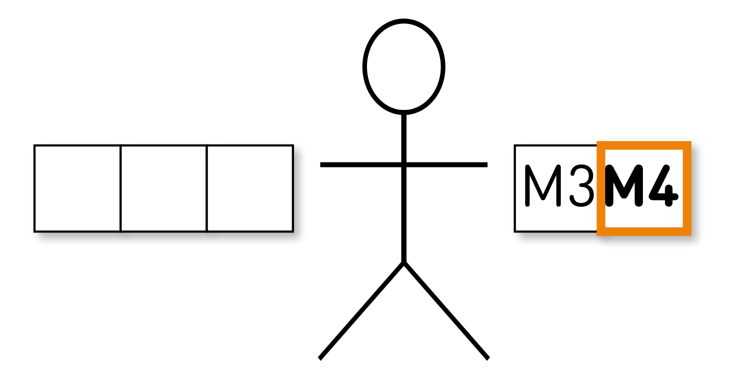


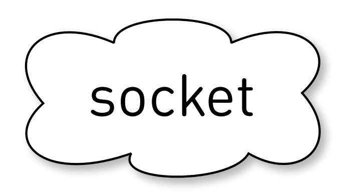

M3

socket



M4
M3


socket




M3
M4


socket

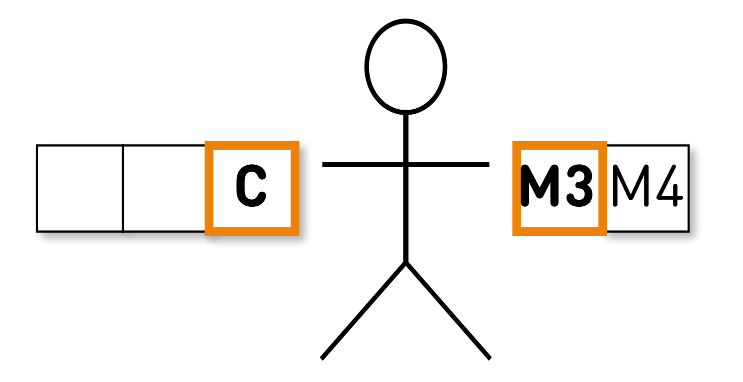
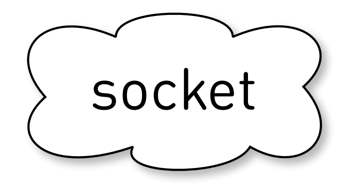


C

M3 M4

socket



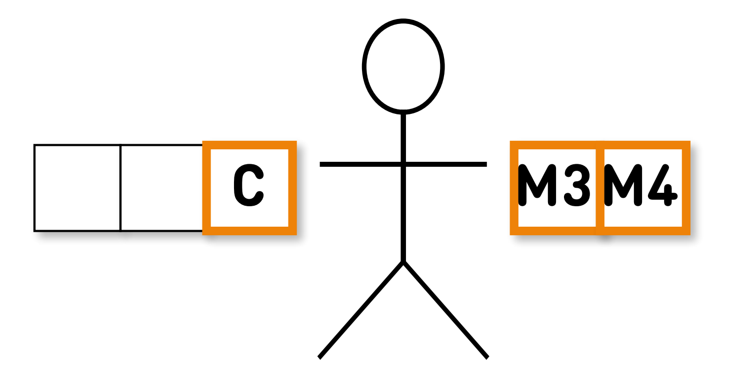

C
M3
M4

socket

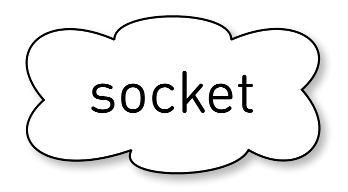

M4

write(M3)

socket




C
M3
M4


socket



write(M3+M4)

socket



SendingBenchmark

- **7 μs/msg**, если по одному



SendingBenchmark

- **7 μs/msg**, если по одному
- **1 μs/msg**, если батчами по **128 КБ**



SendingBenchmark

- **7 μs/msg**, если по одному
- **1 μs/msg**, если батчами по **128 КБ**
- Читаем **тоже батчами** через offheap в `In`



PingPongBenchmark

- **1 M** сообщений
- Каждое **туда и обратно**
- **Последовательно**



PingPongBenchmark

- **1 M** сообщений
- Каждое **туда и обратно**
- **Последовательно**





Let's go deeper

```
void enqueue(M message) {
    messages.offer(message);
    tryScheduleToExecute();
}
```



Even deeper

```java
void tryScheduleToExecute() {
    if (on.compareAndSet(false, true)) {
        try {
            dispatcher.execute(this);
        } catch (Exception e) {
            ...
        }
    }
}
```



Messaging Service

Fabric

accept()

Node 10.0.0.1

Node 10.0.0.2

Node ...

Connection ...

Connection ACK

Connection CMD

Connection ...

In

Out

In

Out

socket

socket



Node
10.0.0.2

Connection
CMD

Thread 1
Thread 2

Out

socket



Node
10.0.0.2

Connection
CMD

Thread 1
Thread 2

Out

socket



Node
10.0.0.2

Connection
CMD

Out

Thread 1

Thread 2

socket



Node
10.0.0.2

Thread 1

Thread 2

Connection
CMD

socket

Out



Node
10.0.0.2

Connection
CMD

Out

Thread 1
Thread 2

socket



Node
10.0.0.2

Connection
CMD

Thread 1

Thread 2

Out

socket



Node
10.0.0.2

Connection
CMD

Thread 1
Thread 2

Out

socket



Node
10.0.0.2

Connection
CMD

Thread 1
Thread 2

Out

socket



Node
10.0.0.2

Connection
CMD

Out

Thread 1

Thread 2

socket



Node
10.0.0.2

Connection
CMD

Thread 1

Thread 2

socket

Out



Node
10.0.0.2

Connection
CMD

Out

Thread 1
Thread 2

socket



Node
10.0.0.2

Thread 1
Thread 2

Connection
CMD

socket

Out



Node
10.0.0.2

Connection
CMD

Out

Thread 1
Thread 2

socket



Call stack вместо context switch?



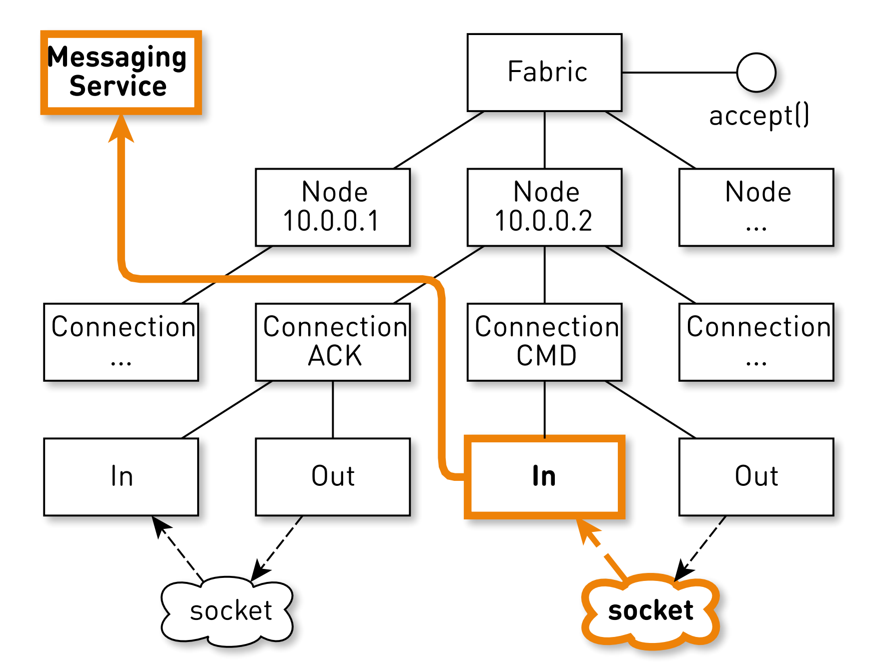
Call stack вместо context switch?

```
@Override
public void execute(Runnable command) {
    Thread current = Thread.currentThread();
    if (current instanceof DispatcherThread) {
        // Use the call stack instead of context switch
        command.run();
    } else {
        super.execute(command);
    }
```



```
10:28:17,508 org.apache.cassandra.service.CassandraDaemon$2 Exception in thread Thread[Fabric:17007:51,5,main]
java.lang.StackOverflowError
        at sun.nio.ch.AsynchronousSocketChannelImpl.read(AsynchronousSocketChannelImpl.java:295)
        at org.apache.cassandra.net.async.connection.In$1.completed(In.java:166)
        at org.apache.cassandra.net.async.connection.In$1.completed(In.java:154)
        at sun.nio.ch.Invoker.invokeUnchecked(Invoker.java:126)
        at sun.nio.ch.Invoker.invokeDirect(Invoker.java:157)
        at sun.nio.ch.UnixAsynchronousSocketChannelImpl.implRead(UnixAsynchronousSocketChannelImpl.java:553)
        at sun.nio.ch.AsynchronousSocketChannelImpl.read(AsynchronousSocketChannelImpl.java:276)
        at sun.nio.ch.AsynchronousSocketChannelImpl.read(AsynchronousSocketChannelImpl.java:297)
        at org.apache.cassandra.net.async.connection.In$1.completed(In.java:166)
        at org.apache.cassandra.net.async.connection.In$1.completed(In.java:154)
        at sun.nio.ch.Invoker.invokeUnchecked(Invoker.java:126)
        at sun.nio.ch.Invoker.invokeDirect(Invoker.java:157)
        at sun.nio.ch.UnixAsynchronousSocketChannelImpl.implRead(UnixAsynchronousSocketChannelImpl.java:553)
         ...
```




Messaging Service
Fabric
accept()
Node 10.0.0.1
Node 10.0.0.2
Node ...
Connection ...
Connection ACK
Connection CMD
Connection ...
In
Out
In
Out
socket
socket

sun.nio.ch.Invoker

```java
/**
 * Returns true if the current thread is in
 * the given channel's thread pool and
 * we haven't exceeded the maximum number of
 * handler frames on the stack.
 */
static boolean mayInvokeDirect(
    GroupAndInvokeCount myGroupAndInvokeCount,
    AsynchronousChannelGroupImpl group) {
```

Ограничиваем call stack



Ограничиваем call stack

```
@Override
public void execute(Runnable command) {
    Thread current = Thread.currentThread();
    if (current instanceof DepthCountingThread) {
        DepthCountingThread thread = (DepthCountingThread) current;
        int depth = thread.depth;
        if (depth == throughput) {
            super.execute(command);
        } else {
            thread.depth = depth + 1;
            try {
                // Use the call stack instead of context switch
                command.run();
            } finally {
                thread.depth = depth;
            }
        }
```

```java
class DepthCountingThread extends Thread {
    int depth = 0;

    DepthCountingThread(
            Runnable target,
            String name) {
        super(target, name);
    }
}
```

PingPongBenchmark RTT

- **54** μs было



PingPongBenchmark RTT

- **54** μs было
- **47** μs при `throughput == 1`



PingPongBenchmark RTT

- **54** μs было
- **47** μs при `throughput == 1`
- **38** μs при `throughput == 4`



Throughput



Throughput

- **64 КБ send**: > 20 Gbps и > 40 Kmps



Throughput

- **64 КБ send**: > 20 Gbps и > 40 Kmps
- **64 Б send**: > 1 Gbps и > 1 Mmps



Throughput

- **64 КБ send**: > 20 Gbps и > 40 Kmps
- **64 Б send**: > 1 Gbps и > 1 Mmps
- **64 КБ ping-pong**: > 14 Gbps и > 15 Kmps

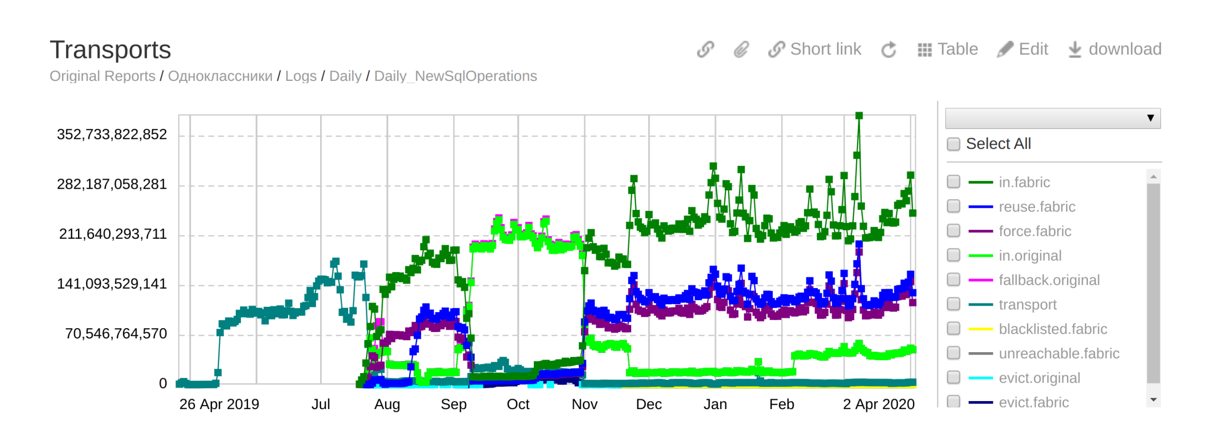


Throughput

- **64 КБ send**: > 20 Gbps и > 40 Kmps
- **64 Б send**: > 1 Gbps и > 1 Mmps
- **64 КБ ping-pong**: > 14 Gbps и > 15 Kmps
- **64 Б ping-pong**: > 80 Mbps и > 30 Kmps



Раскатка в production



Грабли

Грабли

Video DAO Duration Avg.



.Dashboards / Одноклассники / Video / Video Monitoring / DAO / Video DAO Duration Avg

29,343,425
24,636,428
19,929,432
15,222,435
10,515,439
5,808,442
1,101,446

02:00 04:00 06:00 08:00 10:00 12:00 14:00 16:00 18:00 20:00 22:00 24:00

```
enum Checkpoint {
    INIT,       // NEVER USE OR DELETE
    DEBUG,      // Used only for unit tests
    ACCEPTED,   // Accepted for sending
    ENQUEUED,   // Enqueued for sending
    BATCHED,    // Became part of a batch to be sent
    SENDING,    // Just before sending
    SENT,       // Just having sent
    RECEIVED,   // At the destination
    DELIVERING, // Before delivering to MessagingService
    RESPONDING; // MessagingService responds to request
```



Трассировка

- Начальный `timestamp`



Трассировка

- Начальный $\mathtt{timestamp}$
- Список $(\mathtt{Checkpoint}, \Delta t)$



Трассировка

- Начальный `timestamp`
- Список $(\mathtt{Checkpoint}, \Delta t)$
- Разрешение Δt **1 мс**, **max 16 с**
- Δt как `VarUint`



Трассировка

- Начальный `timestamp`
- Список $(\texttt{Checkpoint}, \Delta t)$
- Разрешение Δt **1 мс**, **max 16 с**
- Δt как `VarUint`
- **Всегда сериализована** в `byte[]`
- **1-3 байта** на `Checkpoint`
- **Несколько десятков байт** на трассу



Нежданчик



Нежданчик

- Периодически **200 мс** от `SENDING` до `RECEIVED`



Нежданчик

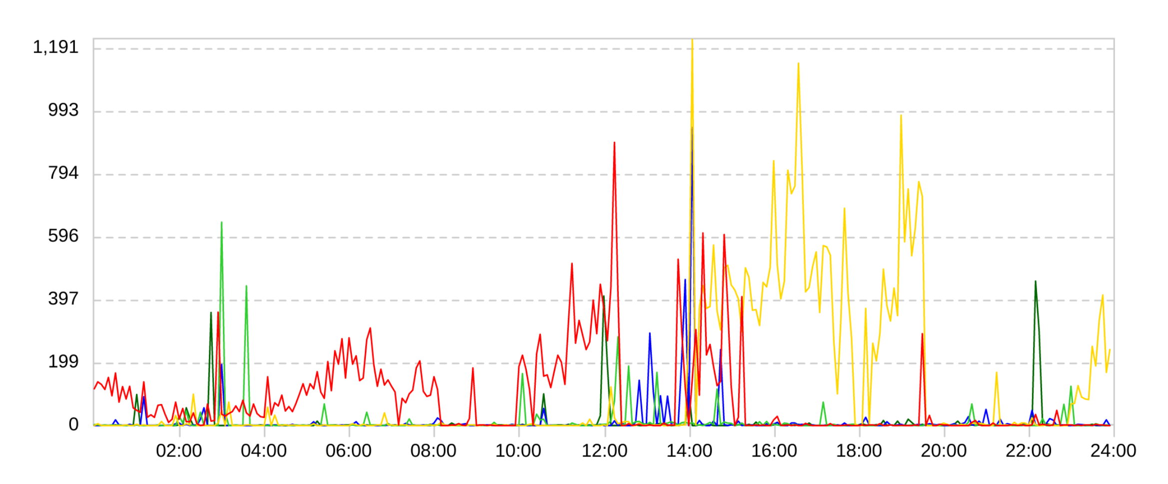
- Периодически **200 мс** от `SENDING` до `RECEIVED`

Нужен `TCP_NODELAY` после `accept()`

Входящие соединения теперь используются для **исходящих сообщений**.



Флапы координаторов каждые 1/5 мин



Включили логи safepoint

- -XX:+PrintGCApplicationStoppedTime
- -XX:+PrintSafepointStatistics
- -XX:PrintSafepointStatisticsCount=1



Включили логи safepoint

- -XX:+PrintGCApplicationStoppedTime
- -XX:+PrintSafepointStatistics
- -XX:PrintSafepointStatisticsCount=1

Achtung!

- Пачки событий (Bulk)RevokeBias
- **Десятки пауз** по **десятку миллисекунд**

Eliminating Synchronization-Related Atomic Operations with Biased Locking and Bulk Rebiasing

Kenneth Russell
Sun Microsystems, Inc.
kenneth.russell@sun.com

David Detlefs *
david.detlefs@alum.mit.edu

Abstract

The Java™ programming language contains built-in synchronization primitives for use in constructing multithreaded programs. Efficient implementation of these synchronization primitives is necessary in order to achieve high performance.

Recent research [9, 12, 10, 3, 7] has focused on the run-time elimination of the atomic operations required to implement object monitor synchronization primitives. This paper describes a novel technique called *store-free biased locking* which eliminates all synchronization-related atomic operations on uncontended object monitors. The technique supports the bulk transfer of object ownership from one thread to another, and the selective disabling of the optimization where unprofitable, using epoch-based bulk rebiasing and revocation. It has been implemented in the production version of the Java HotSpot™ VM and has yielded significant performance improvements on a range of benchmarks and applications. The technique is applicable to any virtual machine-based programming language implementation with mostly block-structured locking primitives.

upon monitor entry, and sometimes upon exit, to ensure correct synchronization. These techniques fall back to using OS mutexes and condition variables when contention occurs.

A related class of optimizations which can be termed *biased locking* [3, 7, 9] rely on the further property that not only are most monitors uncontended, they are only entered and exited by one thread during the lifetime of the monitor. Such monitors may be profitably *biased* toward the owning thread, allowing that thread to enter and exit the monitor without using atomic operations. If another thread attempts to enter a biased monitor, even if no contention occurs, a relatively expensive *bias revocation* operation must be performed. The profitability of such an optimization relies on the benefit of the elimination of atomic operations being higher than the penalty of revocation.

Current refinements of biased locking techniques [12, 10] decrease or eliminate the penalty of bias revocation, but do not optimize certain synchronization patterns which occur in practice, and also impact peak performance of the algorithm.

Multiprocessor systems are increasingly prevalent; so much so that uniprocessors are now the exception rather than the norm.



Biasable Mode

Normal Mode
(not biasable)

Object allocation

Biasable, unbiased

| 0 | 1 | 01 |

initial lock

Biased toward thread T

| T | 1 | 01 |

bulk rebias
(effective transition)

revoke bias

or

| 0 | 01 |

currently unlocked

lock

unlock

| stack pointer | 00 |

currently locked

Figure 2. State transitions of an object's mark word under biased locking.



Отпрофилировали[3] STW-операции

```
-e VMThread::execute --cstack dwarf
```

[3] https://github.com/jvm-profiling-tools/async-profiler

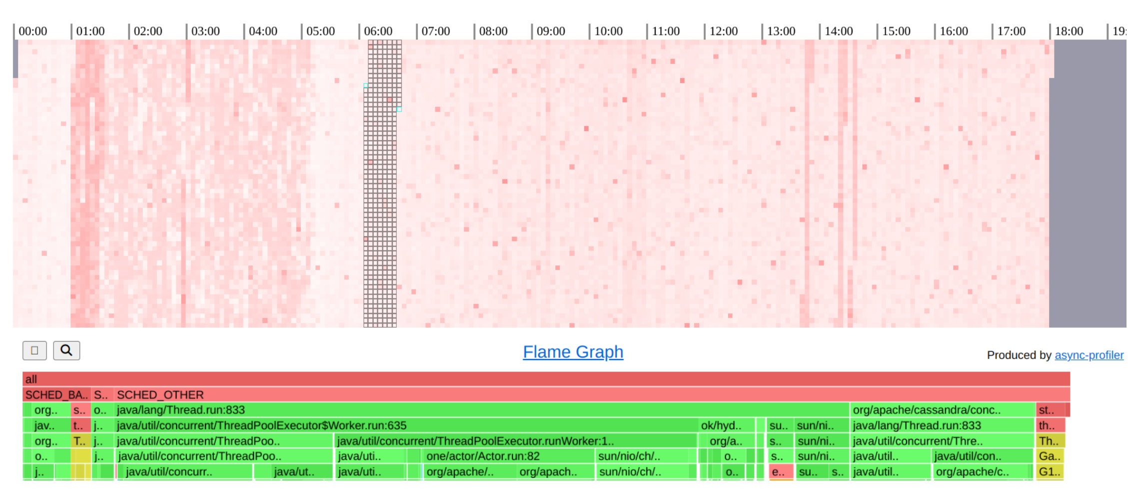
Отпрофилировали[3] STW-операции

```
-e VMThread::execute --cstack dwarf
```

BiasedLocking::revoke_and_rebias(Handle, bool, Thread*)
ObjectSynchronizer::fast_enter(Handle, BasicLock*, bool, Thread*)
SharedRuntime::complete_monitor_locking_C(oopDesc*, BasicLock*, JavaThread*)
javax/management/ObjectName._getKeyPropertyList
javax/management/ObjectName.getKeyProperty
com/sun/jmx/mbeanserver/Repository$ObjectNamePattern.matchKeys
com/sun/jmx/mbeanserver/Repository.addAllMatching
com/sun/jmx/mbeanserver/Repository.query
com/sun/jmx/interceptor/DefaultMBeanServerInterceptor.queryNamesImpl
com/sun/jmx/interceptor/DefaultMBeanServerInterceptor.queryNames
com/sun/jmx/mbeanserver/JmxMBeanServer.queryNames
com/sun/jmx/remote/security/MBeanServerAccessController.queryNames

[3] https://github.com/jvm-profiling-tools/async-profiler

Сейчас ещё проще[4]



[4] Непрерывное профилирование в облаке с помощью eBPF @ JPoint 2022

-XX:-UseBiasedLocking



-XX:-UseBiasedLocking

1. JMX-монитор ходит в сервис **раз в минуту**



-XX:-UseBiasedLocking

1. JMX-монитор ходит в сервис **раз в минуту**
2. one-log отправляет стату **каждые 5 мин**



-XX:-UseBiasedLocking

1. JMX-монитор ходит в сервис **раз в минуту**
2. `one-log` отправляет стату **каждые 5 мин**
3. `one-log` **«трогает»** biased JMX `ObjectName`



-XX:-UseBiasedLocking

1. JMX-монитор ходит в сервис **раз в минуту**
2. `one-log` отправляет стату **каждые 5 мин**
3. `one-log` **«трогает»** biased JMX `ObjectName`
4. Паузы на **сотни миллисекунд** из-за `RevokeBias`



-XX:-UseBiasedLocking

1. JMX-монитор ходит в сервис **раз в минуту**
2. one-log отправляет стату **каждые 5 мин**
3. one-log **«трогает»** biased JMX ObjectName
4. Паузы на **сотни миллисекунд** из-за RevokeBias
5. Координатор считают **мёртвым**



JEP 374: Deprecate and Disable Biased Locking

Owner	Patricio Chilano Mateo
Type	Feature
Scope	JDK
Status	Closed / Delivered
Release	15
Component	hotspot / runtime
Discussion	hotspot dash runtime dash dev at openjdk dot java dot net
Effort	XS
Duration	XS
Reviewed by	Coleen Phillimore, David Holmes, Mikael Vidstedt
Endorsed by	Mikael Vidstedt
Created	2019/12/03 14:24
Updated	2021/08/28 00:39
Issue	8235256

Summary

Disable biased locking by default, and deprecate all related command-line options.



Частые Full GC на стораджах



Частые Full GC на стораджах

```
List<ExpiringMessage> batch = queue.extractBatch();
 ...
out.write (...,
        new CompletionHandler<Integer, ByteBuffer>() {
                @Override
                public void completed(...) {
                        ...
                        // Tracing
                        fillTraceState (batch);
                        ...
```



JDK / JDK-8202252

(aio) Closed AsynchronousSocketChannel keeps completion handler alive

Details

Type:	Bug	Status:	RESOLVED
Priority:	P3	Resolution:	Fixed
Affects Version/s:	7, 8, 10, 11	Fix Version/s:	12
Component/s:	core-libs		
Labels:	additional-information-received azul-interest dcs-pso jdk11u-fix-request jdk11u-fix-yes jdk8u-fix-request jdk8u-fix-yes oracle-bp redhat-interest reproducer-no webbug		
Subcomponent:	java.nio		
Resolved In Build:	b01		
CPU:	x86		
OS:	os_x		

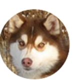


```
        // invoke handler and set result
        CompletionHandler<Void,Object> handler = connectHandler;
+        connectHandler = null;
        Object att  = connectAttachment;
        PendingFuture<Void,Object> future = connectFuture;
        if (handler == null) {
@@ -405,6 +406,7 @@
            this.readBuffer = null;
            this.readBuffers = null;
            this.readAttachment = null;
+            this.readHandler = null;

            // allow another read to be initiated
            enableReading();
@@ -600,6 +602,7 @@
            this.writeBuffer = null;
            this.writeBuffers = null;
            this.writeAttachment = null;
+            this.writeHandler = null;
```

```java
List<ExpiringMessage> batch = queue.extractBatch();
 ...
out.write (...,
      new CompletionHandler<Integer, ByteBuffer>() {
          @Override
          public void completed(...) {
               ...
               // Tracing
               fillTraceState (batch);
               ...
```

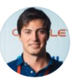
```
List<ExpiringMessage> batch = queue.extractBatch();
...
AtomicReference batchRef = new AtomicReference<>(batch);
out.write (...,
      new CompletionHandler<Integer, ByteBuffer>() {
            @Override
            public void completed(...) {
                  ...
                  // Tracing
                  fillTraceState (batchRef.getAndSet(null));
                  ...
```



Andrei Pangin @AndreiPangin · Jul 16, 2019

Found a severe memory leak in production with @incubos. Turns out to be a bug in JDK Async IO. Ironically the bug is marked as P5 - the lowest priority "cosmetic" problem. No fix until JDK 12. bugs.openjdk.java.net/browse/JDK-820…

1 6 23



Vladimir Ivanov @iwan0www · Jul 16, 2019

1. It's perfectly fine to raise bug priority even after the fix arrived.
2. Aleksey @shipilev might be interested in putting it on 8u & 11u backport list anyway ;-)

1 1 8



Aleksey Shipilëv
@shipilev

Replying to @iwan0www @AndreiPangin and @incubos

In backporting queue for 8u and 11u now.

11:55 PM · Jul 16, 2019 · TweetDeck



Aleksey Shipilev added a comment - 2019-07-16 12:13

Users report this as the "severe memory leak in production" here:
https://twitter.com/AndreiPangin/status/1151174582856011776 -- raising priority.

Aleksey Shipilev added a comment - 2019-07-16 13:51

Fix Request (8u, 11u)

Backporting this fix resolves the serious memory leak in aio. Patch applies cleanly to 11u, and requires usual reshufflings in 8u. New test fails without the product patch (has unusual failure mode: timeouts on failure, when tracked reference is not GC-ed) in both 8u and 11u, and passes with it. Additionally, tier1 passes in 11u and 8u.

Andrei Pangin added a comment - 2019-07-16 14:32

In our case the symptoms are a bit different than in the bug description, but the same fix works fine.

The problem is caused by the writeHandler holding a reference to a CompletionHandler long after the write operation completes. Since we had ~3000 async channels, and each CompletionHandler retained several MB of data, the application exhausted 10 GB heap rather quickly. Setting writeHandler=null in finishWrite() solves the issue.

Andrei Pangin added a comment - 2019-07-21 06:55

Added another test for this bug (AsyncChannelLeak.java), which demonstrates the above scenario.

Соединения не работают и не рвутся



Соединения не работают и не рвутся

Дропаем пакеты:

```
$ iptables -A INPUT -p tcp --dport 65000 -j DROP
$ iptables -A INPUT -p tcp --sport 65000 -j DROP
```



Соединения не работают и не рвутся

Дропаем пакеты:

```
$ iptables -A INPUT -p tcp --dport 65000 -j DROP
$ iptables -A INPUT -p tcp --sport 65000 -j DROP
```

Свежие входящие соединения

Вместо старых **потенциально подвисших**.



И многое другое



И многое другое

- Gossip только **`client`** $\longrightarrow$ **`node`** и `node` $\longleftrightarrow$ `node`



И многое другое

- Gossip только **`client`** $\to$ **`node`** и `node` $\leftrightarrow$ `node`
- Выпилили `EchoMessage`

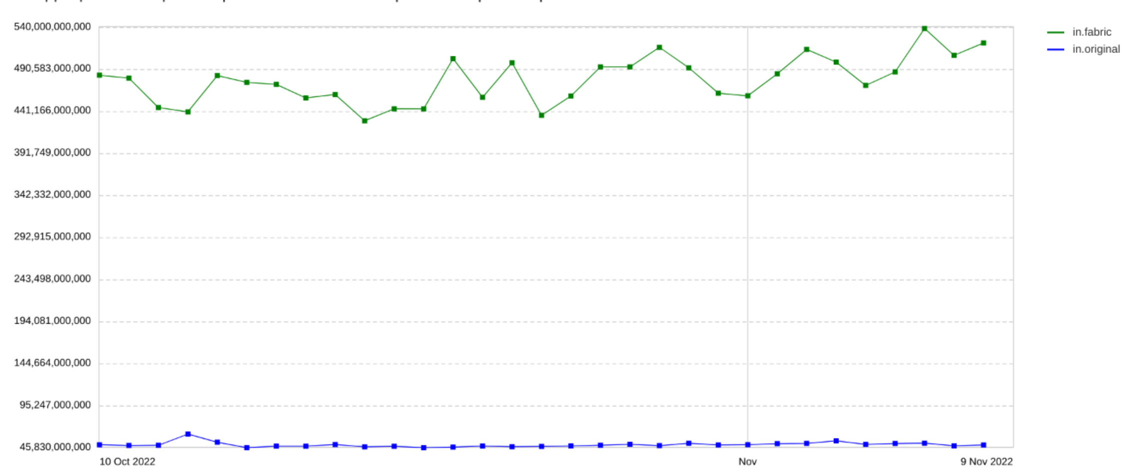


И многое другое

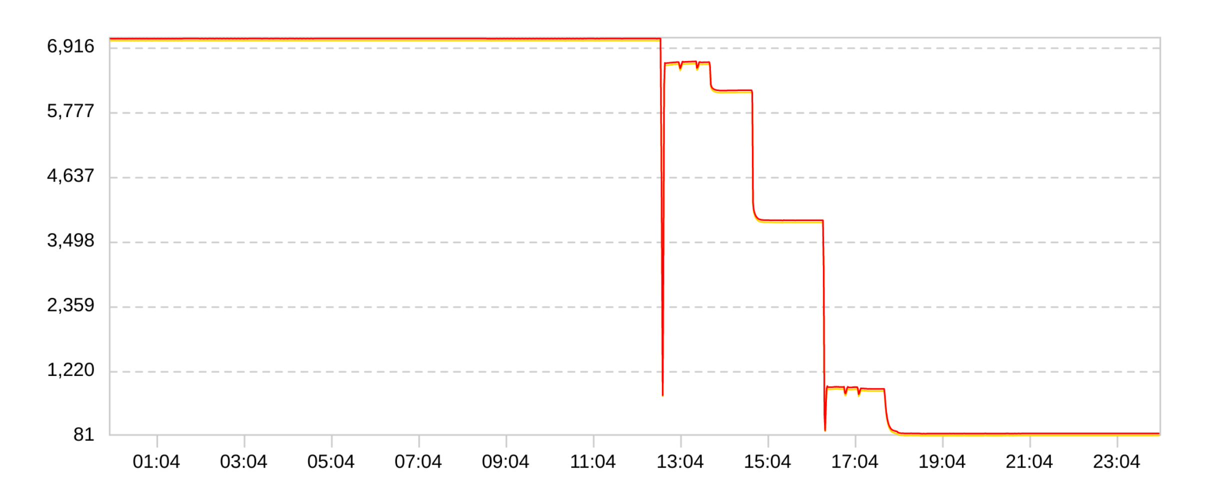
- Gossip только **`client`** $\rightarrow$ **`node`** и `node` $\leftrightarrow$ `node`
- Выпилили `EchoMessage`
- В(ы)ключение транспорта без downtime
- ...



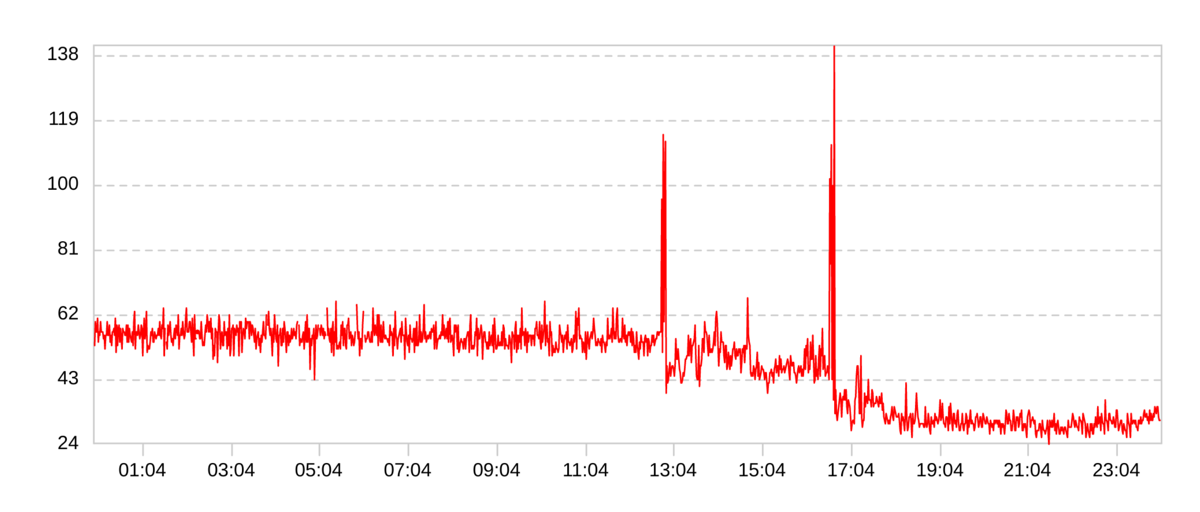
Production


Входящие сообщения оригинального/асинхронного транспорта
540,000,000,000
490,583,000,000
441,166,000,000
391,749,000,000
342,332,000,000
292,915,000,000
243,498,000,000
194,081,000,000
144,664,000,000
95,247,000,000
45,830,000,000
10 Oct 2022
Nov
9 Nov 2022
in.fabric
in.original

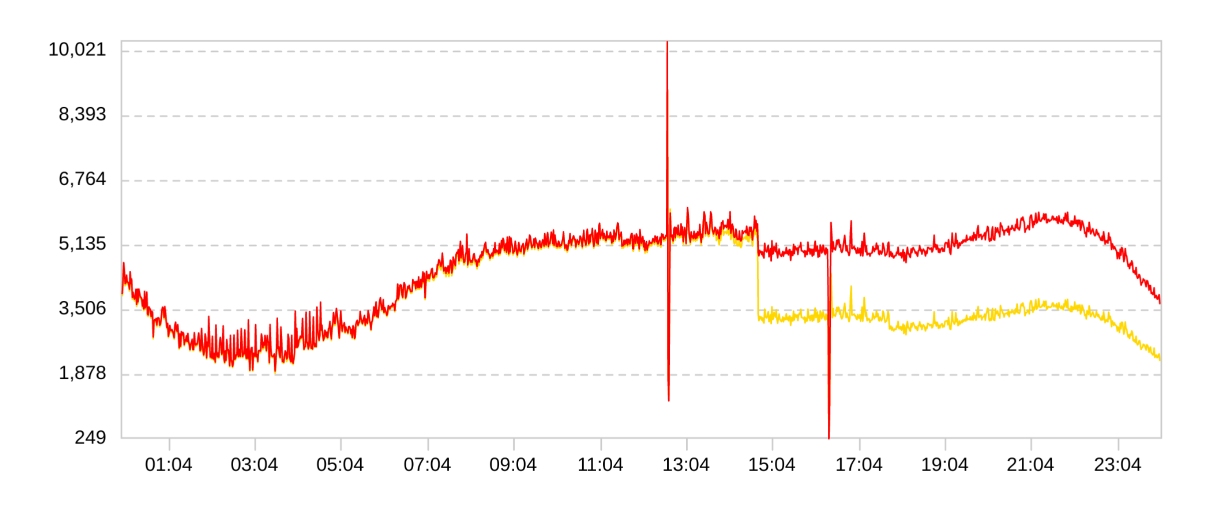
Потоки на ноде



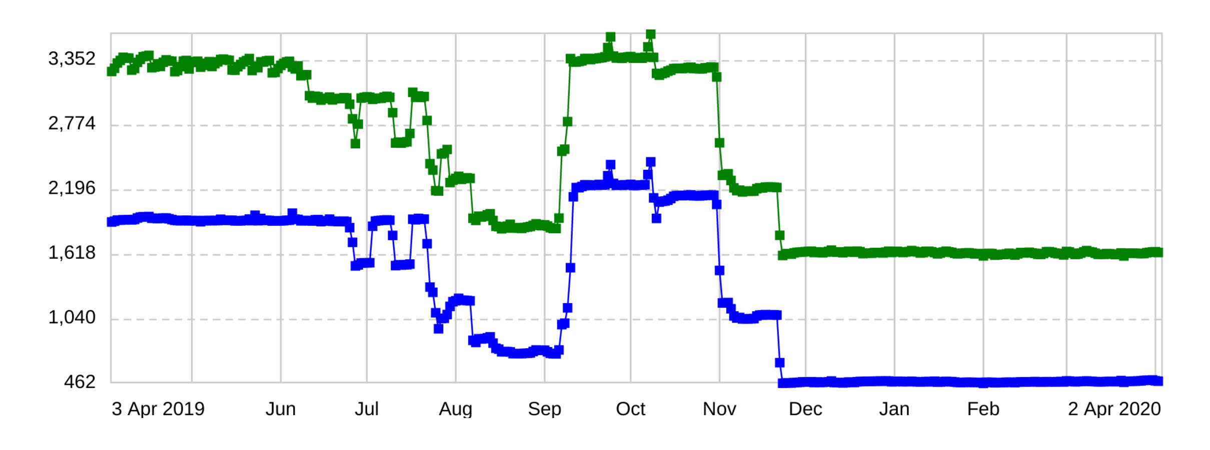
Время GC, мс


138
119
100
81
62
43
24
01:04
03:04
05:04
07:04
09:04
11:04
13:04
15:04
17:04
19:04
21:04
23:04

Сетевые пакеты



Потоки на клиенте



В заключение

- Цели **достигнуты**
  - Меньше потоков
  - Нет соединений к клиентам
  - Улучшили метрики GC



В заключение

- Цели **достигнуты**
  - Меньше потоков
  - Нет соединений к клиентам
  - Улучшили метрики GC
- Хочется **большей производительности**



В заключение

- Цели **достигнуты**
  - Меньше потоков
  - Нет соединений к клиентам
  - Улучшили метрики GC
- Хочется **большей производительности** — полезайте в **корные компоненты**



В заключение

- Цели **достигнуты**
  - Меньше потоков
  - Нет соединений к клиентам
  - Улучшили метрики GC
- Хочется **большей производительности** — полезайте в **корные компоненты**
- **Знайте и любите** свой стек



В заключение

- Цели **достигнуты**
  - Меньше потоков
  - Нет соединений к клиентам
  - Улучшили метрики GC
- Хочется **большей производительности** — полезайте в **корные компоненты**
- **Знайте и любите** свой стек — грабли везде





Cassandra / CASSANDRA-15066

Improvements to Internode Messaging

Details

Type:	Improvement	Status:	RESOLVED
Priority:	High	Resolution:	Fixed
Component/s:	Messaging/Internode	Fix Version/s:	4.0-alpha1, 4.0
Labels:	None		
Change Category:	Quality Assurance		
Complexity:	Challenging		

Async IO



Actor Model

=

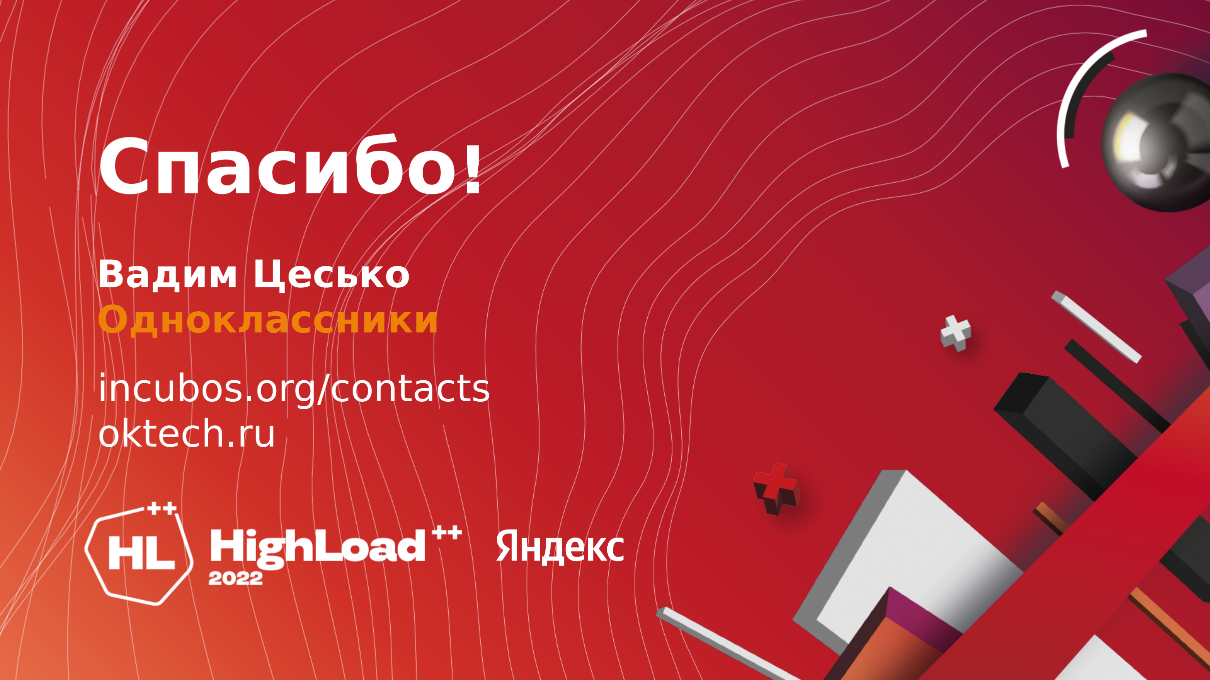




Благодарности

- Александр Христофоров
- Андрей Паньгин
- Олег Анастасьев




Спасибо!
Вадим Цесько
Одноклассники
incubos.org/contacts
oktech.ru
HL
HighLoad++
2022
Яндекс

Обратная связь
и комментарии
по докладу ------------->

Слайды:
http://bit.ly/3glLhRa



